DOCUMENT RESUME

ED 317 190 IR 014 275

AUTHOR	Ryan, William C., Ed.
TITLE	NECC '89. Connections. Proceedings of the National Educational Computing Conference (10th, Boston, Massachusetts, June 20-22, 1989).
SPONS AGENCY	National Educational Computing Conference.
REPORT NO	ISBN-0-924667-61-3
PUB DATE	Jun 89
NOTE	414p.; For the proceedings of the 1988 conference, see ED 304 125.
PUB TYPE	Collected Works - Conference Proceedings (021) -- Viewpoints (120) -- Reports - Research/Technical (143)
EDRS PRICE	MF01/PC17 Plus Postage.
DESCRIPTORS	Artificial Intelligence; Cognitive Processes; *Computer Assisted Instruction; Computer Networks; Computer Simulation; *Courseware; Disabilities; Elementary Secondary Education; Expert Systems; Faculty Development; Higher Education; High Risk Students; *Hypermedia; Instructional Development; *Interactive Video; *Microcomputers; Problem Solving; Programing; *Telecommunications

ABSTRACT

The papers presented at the 1989 National Educational Computing Conference focused on ways of using technology to improve educational quality. Topics of the 50 papers and more than 120 abstracts provided in these proceedings include applications of artificial intelligence and the development of expert systems; authoring systems; using the computer and spreadsheets to develop problem solving and critical thinking skills; computer conferencing; computer networks; computer simulations; teaching programming skills; use of computer technology to assist at-risk students and those with disabilities; computers for math, science, and equity; using the Logo programming language; integrating computers into Star Schools; the Apple Classroom of Tomorrow; inservice training for teachers using computers in the classroom; using word processing programs in teaching writing; videodisk and hypertext (HyperCard) applications; and the use of telecommunications. An author index is included. (GL)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED317190

PROCEEDINGS

National Educational Computing Conference

Hynes Convention Center
Boston, Massachusetts
June 20 - 22, 1989

NECC '89 Connections

Hosted by

Lesley College
in collaboration with:

Bolt Beranek and Newman
Boston Computer Society
Education Development Center
Educational Technology Center
Massachusetts Computer Using Educators
Massachusetts Department of Education
The Computer Museum
Technical Education Research Centers

BEST COPY AVAILABLE

IR014275

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
ICCE

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

PROCEEDINGS

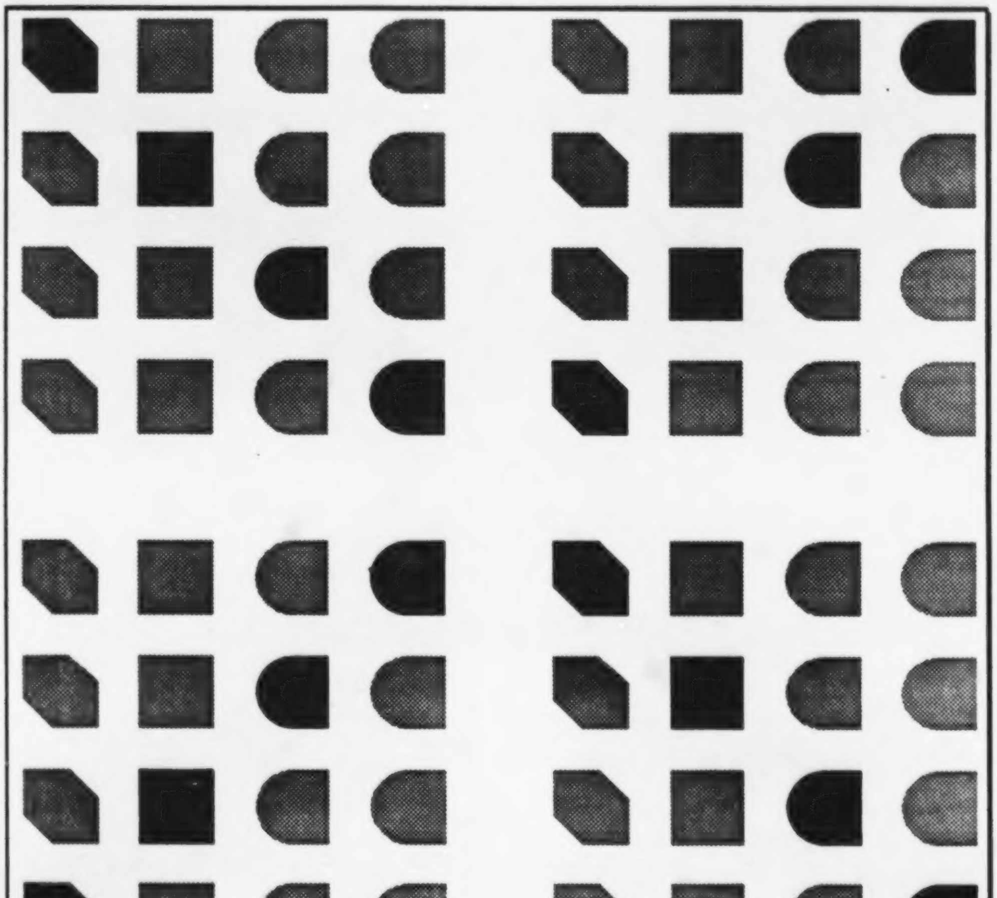
NATIONAL EDUCATIONAL COMPUTING CONFERENCE 1989
BOSTON, MASSACHUSETTS JUNE 20-22

Hosted by

Lesley College

in collaboration with

Bolt Beranek and Newman
Boston Computer Society
Education Development Center
Educational Technology Center
Massachusetts Computer Using Educators
Massachusetts Department of Education
The Computer Museum
Technical Education Research Centers



William C. Ryan, Editor

NATIONAL EDUCATIONAL COMPUTING CONFERENCE

COOPERATING SOCIETIES

Association for Computers and the Humanities (ACH)
Association for Computing Machinery/Special Interest Groups on:
  Computers and Society (SIGCAS)
  Computer Science Education (SIGCSE)
  Computer Science Education (SIGCSE)
  Computer Uses in Education (SIGCUE)
  University and College Computing Services (SIGUCCS)
Association of Small Computer Users in Education (ASCUE)
Educational Computing at Minority Institutions (ECMI)
EDUCOM/BITNIC
International Association for Computing in Education (IACE)
International Council for Computers in Education (ICCE)
IEEE Computer Society
Society for Computer Simulation (SCS)

NECC STEERING COMMITTEE

Doris K. Lidtke
Chair
Towson State University

William C. Ryan
Vice-Chair
Swarthmore College

Ted Sjoerdsma
Treasurer
Washington and Lee University

Diana Harris
Secretary
University of Iowa

Ron Anderson
ACM/SIGCAS
University of Minnesota

Gary Bitter
IACE
Arizona State University

George Brett
ACH
Univeristy of North Carolina

Gerald L. Engel
Past Chair, Steering Committee
University of Connecticut at Stamford

Susan Friel
Co-chair, NECC '89
Lesley College

Steven Gilbert
EDUCOM
Princeton, NJ

James Kerlin
ACM/SIGUCCS
Pennsylvania State University

Barbara Kurshan
ICCE
Roanoke, VA

John Lawson
ACM/SIGCUE
Towson State University

Jesse C. Lewis
ECMI
Norfolk State University

John McGregor
Conference Chair, NECC '90
Murray State University

Dave Moursund
ICCE
University of Oregon

Michael Mulder
IEEE Computer Society
University of Portland

Cathleen Norris
Program Chair, NECC '88
Univeristy of North Texas

Jim Poirot
Chair, NECC '88
University of North Texas

John Richards
Program Chair, NECC '89
Bolt Beranek and Newman

Nancy Roberts
SCS; Co-chair, NECC '89
Lesley College

W. Waldo Roth
ASCUE
Taylor University

Thomas Saterfiel
AERA
Mississippi State Department of Education

A. Joe Turner
ACM/SIGCSE
Clemson University

PROCEEDINGS OF NECC '89

International Standard Book Number: 0-924667-61-3

Copyright © 1989: NECC

Published by:
International Council on Computers for Education
University of Oregon
1787 Agate Street
Eugene, OR 97403-1923
for:
National Educational Computing Conference 1989
June 1989

NECC '89

NECC '89 CONFERENCE COMMITTEE

Susan Friel
Conference Co-chair
Lesley College

Nancy Roberts
Conference Co-chair
Lesley College

John Richards
Program Chair
Bolt Beranek and Newman

Angeline Ferris
Treasurer
Lesley College

Evelyn Woldman
Registrar
Framingham, MA

Michael Battista
Projects
Kent State University

Douglas Clements
Projects
New York State University at Buffalo

Betty Bjork
Local Arrangements
Education Development Center

Sharon Finke
Exhibits
University of Oregon

Paul Katz
Exhibits
University of Oregon

Anne Knight
Workshops
Harvard University

Beth Lowd
Publicity
Lexington Public Schools

Alan November
Social Events
Wellesley Public Schools

Susan Jo Russell
Papers
Technical Education Research Centers

William Ryan
Proceedings
Swarthmore, PA

A. Joe Turner
Society Sessions
Clemson University

LOCAL ARRANGEMENTS AND PROGRAM ACTIVITIES

Gerri Abrams
Wilmington Public Schools

Jim Adams
Association for Computing Machinery

Linda Colvin
Walpole Public Schools

Joan Dunfey
Lesley College

Theresa Duggan
Concord Public Schools

Susan Foote
Massachusetts Department of Education

Karen Gremley
Lesley College

Chuck Hitchcock
Chapter I

Phyllis Kalowski
Cambridge Public Schools

Diana MacLean
Concord Public Schools

James MacNeil
Concord Public Schools

Jane Manzelli
Brookline Public Schools

Marianne McDonald
Education Development Center

Inabeth Miller
Boston Museum of Science

Adeline Naiman
The Computer Museum

Diana Nunnaley
Massachusetts Chapter I Computer Center

Henry Olds
Learningways, Inc.

Jeff Pankin
Lesley College

Marilyn Quinsaat
Bank Street College of Education

Gail Ross-McBride
The Education Cooperative

Algot Runeman
King Philip Regional High School

Cindy Stephens
D. C. Heath

Robert Tucker
Chapter I

Joan Warren
Arlington Public Schools

Molly and Dan Watt
Education Development Center

Stone Wiske
Educational Technology Center

ASSOCIATION FOR COMPUTERS AND THE HUMANITIES

The Association for Computers and the Humanities (ACH) is an international organization devoted to encouraging the development and use of computing techniques in humanities research and education. Traditionally, ACH has fostered computer-aided research in literature and language, history, philosophy, anthropology, and related social sciences as well as computer use in the creation and study of art, music, and dance. As computing applications in the humanities have developed and broadened in the 1980s, the Association has expanded its scope to include areas from word processing to computer-assisted instruction in composition, language, history, philosophy, and anthropology, as well as computational linguistics and cognitive science, which overlap increasingly with work in the area of humanities computing.

Founded in 1977, ACH is the primary professional society for humanists who are involved or interested in any aspect of humanities computing. The Association provides a forum for continuing communication about humanities computing and strives to meet the needs of those who want to gain familiarity with both existing and potential applications of computers in humanities disciplines.

The heart of ACH is its quarterly newsletter, which covers the activities of the Association and its members and includes articles on various areas within humanities computing, news of projects and conferences of interest to ACH members, and reports on activities of governmental agencies and other organizations that affect computer-aided humanities research.

Computers and the Humanities, published by Paradigm Press, is a quarterly journal devoted to scholarship in the field of humanities computing. Subscription to *CHum* is included in the ACH membership fee.

ACH sponsors the bi-annual International Conference on Computers and the Humanities (ICCH), held in odd-numbered years, which brings together scholars from around the world to report on research activities and software and hardware developments in the field. Recently, ACH began to sponsor conferences and workshop on specialized topics in humanities computing, held in even-numbered years.

ACM SPECIAL INTEREST GROUP ON COMPUTERS AND SOCIETY

SIGCAS is the Association for Computing Machinery's Special Interest Group on Computers and Society. With a membership of nearly 1,200, this professional group seeks to identify social issues raised by computer technology and to provide a forum for discussion on how to apporach these issues.

SIGCAS publishes a quarterly newsletter, *Computers and Society*, which is a primary source of material on this topic. As a vehicle of communication for the SIGCAS membership, it includes news, comments, and articles on any societal issues raised by computing technology. One of the few periodicals on this subject, it provides a flexible and timely forum for important, evolving topics, such as data quality, employment, and intellectual property rights.

According to a recent membership survey, about 40% of SIGCAS members teach course material on computers and society. In recent years SIGCAS has organized sessions at computer conferences on topics such as computer ethics and organizational impacts of computers. A special conference on "Computers and the Quality of Life" is being developed and will be held in September, 1990.

ACM SPECIAL INTEREST GROUP ON COMPUTER SCIENCE EDUCATION

SIGCSE became a special interest group of ACM in 1970. It currently consists of over 2000 members from the educational, industrial, and governmental communities interested in various aspects of computer science education. SIGCSE has goals of encouraging and assisting in the development of effective academic programs and courses in computer science and promoting research in computer science education.

The following are objectives of SIGCSE:

1. To provide a continuing forum for the discussion of common problems among educators and other computer scientists through organized meetings and symposia.
2. To publish a bulletin at least quarterly containing information aimed specifically at those interested in computer science education.
3. To work closely with the Education Board of ACM to insure implementation of effective education programs by the Association.

ACM SPECIAL INTEREST GROUP ON COMPUTER USES IN EDUCATION

The Special Interest Group on Computer Uses in Education (SIGCUE) of the Association for Computing Machinery (ACM) provides a forum for the discussion of ideas, methods, and policies related to all aspects of computers in the educational process. Established in 1969 its membership (over 1,500 persons) comes from many countries and numerous, diverse institutions.

SIGCUE publishes a newsletter titled the SIGCUE *Outlook*. SIGCUE also sponsors and organizes technical sessions at ACM annual meetings, the National Educational Computing Conference, and other national and regional meetings of interest to its members. A recent SIGCUE project consisted of a study group on national educational computer policy issues. This culminated in the publication by ACM of an edited volume entitled *National Educational Computer Policy Alternatives*. During the past year, SIGCUE cosponsored a project with the ACM Education Board and the International Council for Computers in Education a project that resulted in a model preservice course in educational computing for elementary education students. The background papers were published in the Summer '88 issue of *OUTLOOK*.

Among SIGCUE's goals are (1) publishing a current, substantive bulletin, (2) cooperating with other special interest groups or educational societies to promote attention to educational computing issues, and (3) helping to bring the technical expertise within ACM to bear upon educational computing generally.

ACM SPECIAL INTEREST GROUP ON COLLEGE & UNIVERSITY COMPUTING SERVICES

SIGUCCS provides a forum for those involved in providing computing services on a college or university campus. The topics addressed by SIGUCCS include managing campus computing, computing as it relates to the overall goals of the institution, and the state-of-the-art in various types of college and university computing services. Membership in SIGUCCS will help an individual to keep current with the world of collegiate computing services, and provides opportunities to discuss and share ideas and experiences with others.

Two annual conferences are regular activities of SIGUCCS. The Computer Center Management Symposium addresses the many aspects of managing computing on campus. This includes hardware, software, planning, finances, and personnel, to name a few. The User Services Conference deals more directly with the delivery of particular services to the higher education community. Tutorials on relevant issues are held at both conferences.

In other projects, SIGUCCS offers a Peer Review of the university computing function. Upon request of the computer center director, members of SIGUCCS will formally analyze and comment on different areas of your campus computing function. SIGUCCS also publishes a quarterly Newsletter. We consider the Newsletter our most important form of communication as it reaches all members and is subscribed to by numerous university computing centers. Conference proceedings are published either as separate documents or as part of the Newsletter itself.

THE ASSOCIATION OF SMALL COMPUTER USERS IN EDUCATION

The Association of Small Computer Users in Education (ASCUE) is a group of small colleges who work toward continued quality computer education and computer assistance in both administrative and academic areas. The purposes of the organization are: 1) to encourage appropriate uses of computing equipment and techniques for the improvement of its member institutions; 2) to supply its members with information on the most current computing methods; 3) to assist them in solving individual problems; 4) to cooperate with them in the utilization of the various small computers; 5) to cooperate with manufacturers, distributors, and suppliers in establishing and maintaining proper technical standards and in meeting new needs for special devices and systems.

EDUCATIONAL COMPUTING IN MINORITY INSTITUTIONS

ECMI is an organization run by a steering committee representing institutions whose student body reflects a large identifiable minority population. The objectives of ECMI are:

1. Computer literacy: To create among the faculty and administrators of the minority institutions an awareness and understanding of the strengths and weaknesses, uses and misuses, advantages and disadvantages, feasibility, practicability and limitations of computer applications in all aspects of society, including education.

2. Educational Computing: To narrow the gap which exists between the faculties in minority and non-minority institutions with respect to educational computing know-how and access.

3. Research Computing: To improve the computing facilities available to faculty of minority institutions for research purposes, particularly in those institutions offering graduate programs.

4. Technical assistance—consultants: To provide expert and impartial technical assistance to academic administrators of minority institutions on all phases of academic computing (instruction and research).

5. Education programs in the computer sciences: To improve the offerings of courses and degree programs in the computer sciences at minority institutions at all levels (e.g., introductory courses, minors, 2-year degree programs, 4-year degree programs, continuing education, graduate programs).

6. Computing facilities: To improve both quality and quantity of computing facilities available at minority institutions, because experience in the non-minority institutions has shown that an adequately staffed and equipped computer center for academic computing is essential to the success of previously stated objectives.

7. Direct student assistance: To increase the availability of minority staff for the computer centers and computer science education programs of minority institutions.

8. The need for a comprehensive program: To facilitate co-ordination and equitable distribution of funded activities to qualified institutions, associations, etc.

EDUCOM

EDUCOM is a nonprofit consortium of colleges, universities, and other institutions founded in 1964 to facilitate the introduction, use, and management of information technology. Through direct services and cooperative efforts, EDUCOM assists its members and provides leadership to the higher education community.

EDUCOM is funded by membership dues, service fees, and grants from foundations, corporations, and government. EDUCOM activities involve over 530 institutions and 90 corporations in the U.S. and abroad. In brief, they are: EDUCOM General Membership; the EDUCOM Consulting Group; EDUCOM Networking Activities including the Networking and Telecommunications Task Force (NTTF) and BITNIC, the BITNET Network Information Center; the EDUCOM Software Initiative; and the Corporate Associates Program.

For information on EDUCOM please write to P.O. Box 364, 777 Alexander Road, Princeton, New Jersey, 08540; 609 520-3340; or for general membership information via BITNET send to: MEMBER@EDUCOM.

INTERNATIONAL ASSOCIATION FOR COMPUTING IN EDUCATION

The International Association for Computing in Education (IACE), formerly the Association for Educational Data Systems (AEDS), continues to serve the educational computing field in three major areas: administration, instruction, and research. It is developing international ties with the ministries of education of six nations and also expanding its individual membership internationally. In addition, there is an increased emphasis on the administrative and instructional uses of microcomputers in higher education. Having weathered recent financial difficulties with the help of a grant from Epson America, Inc., the backlog of issues of the *Journal of Research on Computing in Education* has been produced, and the publication of the IACE Newsletter has resumed. In addition, IACE has entered into a management arrangement with the American Educational Research Association (AERA) to greatly reduce administrative costs. As a result of this arrangement, IACE will have its future annual conferences in conjunction with the AERA annual meetings, and the first IACE/AERA co-conference will be held April 5-9, 1988 in New Orleans. The theme of that conference will be Educational Computing: Practice, Policy, and Research. With the name change, new management arrangement with AERA, new membership structure, and new areas of emphasis, IACE will continue its 26-year tradition of serving the field of educational computing and looks forward to establishing cooperative efforts with other associations in the field.

THE INTERNATIONAL COUNCIL FOR COMPUTERS IN EDUCATION

The International Council for Computers in Education, ICCE, is a non-profit educational organization, with 12,000 Individual Members and 62 Organization and Associate Members.

ICCE publishes *The Computing Teacher*, C.A.L.L. Digest, and books and courseware for persons interested in the instructional use of computers at the precollege level. These publications emphasize teaching about computers, teaching using computers, teacher education, and the impact of computers on curriculum.

ICCE has a substantial and growing professional outreach program. Five major components currently include:

1. Organization and Associate Members. Members publish newsletters and/or journals, hold conferences, and directly interact with their own members.

2. Professional Staff. ICCE has a professional staff who write, edit, participate in conferences, process orders, consult by phone or mail, etc. *The Computing Teacher* is put together by a full in-house production staff.

3. Ad Hoc Committees. Such a committee recently created the "ICCE Policy Statement on Software Copyright" and "Code of Ethical Conduct for Computer Using Teachers".

4. Special Interest Groups. ICCE has organized special interest groups for computer coordinators, teachers of educators, computer science educators, and Logo-using educators.

5. Independent Study Courses. ICCE offers seven independent study courses carrying graduate credit from the Oregon State System of Higher Education.

THE IEEE COMPUTER SOCIETY

The Computer Society is the world's largest association of computing professionals, with a total membership of approximately 100,000 computer scientists, computer engineers, and allied professionals. Society membership is open to IEEE members, associate members, and student members and to non-IEEE members who qualify for affiliate membership. An affiliate member is a person who has achieved status in his or her chosen field of specialization and whose interests focus in the computing field.

Every Computer Society member receives *COMPUTER*, a peer-reviewed monthly magazine of general interest to computing professionals which also covers society news and events. Five specialized magazines and three journals are also available to society members as optional subscriptions and to nonmembers, libraries, and organizations.

Magazines published by the Computer Society include *IEEE COMPUTER GRAPHICS & APPLICATIONS, IEEE MICRO, IEEE DESIGN & TEST, IEEE SOFTWARE*, and *IEEE EXPERT*. Research-oriented journals include *IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, and *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*. The Computer Society Press publishes nonperiodical literature, including tutorial texts and conference records. Last year 95 new titles were produced, and the society's catalog now contains almost 800 titles.

The society sponsors or cosponsors more than 100 conferences and meetings ranging from workshops and symposia with a few dozen participants to major conferences with many thousands of attendees. Over 30 technical committees offer the opportunity to interact with peers in technical specialty areas, receive newsletters, and conduct conferences and tutorials.

The Computer Society has over 100 local chapters throughout the world, and an additional 100-plus student chapters which provide the opportunity to interact with local colleagues and hear experts discuss technical issues. In addition, tutorials, educational activities, accreditation of computer science and engineering academic programs, the development of standards, and an international electronic mail network all play prominent roles in the society's activities.

THE SOCIETY FOR COMPUTER SIMULATION

SCS is the only technical society devoted primarily to the advancement of simulation and allied technology. It has a worldwide membership and a network of regional councils that covers the United States, Canada, and the United Kingdom.

Simulation is used in every scientific and technical discipline including engineering, manufacturing, biomedical, business, and aerospace. Artificial intelligence, robotics, and CAD/CAM and simulators are areas that have been specifically recognized as important to SCS members.

There are three major SCS conferences each year. They are the SCS Western Multiconference, the SCS Eastern Multiconference, and the Summer Computer Simulation Conference (SCSC).

FOREWORD

This tenth National Educational Computing Conference builds on a twenty year tradition of gathering together educators with an interest in using technology as an aid to teaching and learning. The field of computers in education has gone through amazing changes in the sense of the cost and available equipment, but perhaps less dramatic changes when we look at the teaching/learning process. This year's conference, with its theme of *Connections*, focuses on presenting the very best ideas in the uses of technology to improve educational quality. These *Proceedings* provide written reports on these ideas.

A conference of this size, complexity, and emphasis on technology is an undertaking most of us had not imagined. An enormous thank you goes to the many tens of people who dedicated a year or more of all their *free* time to making this really happen. The core committee, who stood by us in the most tense of times includes our program chair, John Richards; treasurer, Angie Ferris; local arrangements chair, Betty Bjork; Regsistrar, Evye Woldman; and publicity chair, Beth Lowd, and *Proceedings* editor, Bill Ryan.

These chairpersons had in turn the service of very dedicated committees, whose timely completion of tasks made this all happen by June 19. Serving on the program committee were Susan Jo Russell for papers, Mike Battista and Doug Clements for projects, Anne Knight for workshops, Joe Turner for society sessions, Susan Foote for ongoing events, Linda Colvin for presiders, and Dan and Molly Watt for Birds-of-a-feather sessions. On the local arrangements committee Jeff Pankin, equipment director, Karen Gremley, tours coordinator, Jim MacNeil, catering and transportation coordinator, and Marilyn Quinsaat, who arranged the evening at The Computer Museum and the Children's Museum, deserve special acknowledgement. Evye Woldman, as registrar, had the able assistance of Diana Nunnaley, Algot Runeman, Phyllis Kalowski, and Chuck Hitchcock. Angie Ferris was assisted as treasurer by Maureen Brown, and Beth Lowd in publicity by Jim Adams and Inabeth Miller.

The NECC Steering Committee, with Doris Lidtke as chair and Diana Harris as Secretary, played an invaluable consulting role, guiding us in keeping with NECC tradition. The ICCE office with Anita Best as liaison, was invaluable as checker, facilitator each step along the way, and general headquarters site.

As has been the case for the last several years, Paul Katz and Sharon Finke organized the exhibitors' show and presentations without a hitch.

We would like to express our gratitude to the faculty and administration at Lesley College, who helped us whenever we asked.

Finally, a big thank you to all the attendees who came to Boston (in spite of the added expense) whose enthusiasm for the program has wiped away all the *"Why did we volunteer?"* questions we asked so many times.

Susan Friel
Nancy Roberts
Co-Chairs, NECC '89
Lesley College
Cambridge, MA 02138-2790

TABLE OF CONTENTS

TU1-1
1 IN SEARCH OF GENERAL PLANNING SKILLS Sharon M. Carver, Mark L. Walker
10 COMPUTER PROGRAMMING AND PROBLEM SOLVING ABILITIES: A META-ANALYSIS Yuen-Kuang Liao and George W. Bright

TU1-2
18 LX LIVE Sharon (Burrowes) Yoder

TU1-3
19 NEW NATIONAL DATA ON THE USE AND EFFECTIVENESS OF COMPUTER-BASED INSTRUCTION Henry Jay Becker

TU1-4
20 EXPLORING MOTION THROUGH GAME PROGRAMS Eadie Adamson, Michael Tempel
TECHNOLOGY IN MIDDLE SCHOOL PHYSICAL SCIENCE: GENDER EQUITY Trish Koontz
AN INNER-CITY MATH AND SCIENCE ENRICHMENT PROGRAM FEATURING LEGO TC LOGO Gary S. Stager

TU1-5
22 THE ROLE OF THE SUBJECT MATTER EXPERT IN THE DEVELOPMENT OF IVD COURSEWARE Edward C. Beardslee
"SEA MAMMALS": A REPURPOSED INTERACTIVE VIDEODISC-HYPERMEDIA HYPERCARD LESSON FOR K THROUGH 2ND GRADE STUDENTS J. Allen Watson, Carole J. Meshot, Carole S. Nelson

TU1-6
23 ART, COMPUTERS, AND THE DEVELOPING CHILD Nancy Scali

TU1-7
24 THE USE OF COMPREHENSIVE COMPUTER-BASED INSTRUCTION FOR THE BASIC SKILLS REMEDIATION OF EDUCATIONALLY DISADVANTAGED STUDENTS Karen Swan, Marco Mitrani
33 A SURVEY OF COMPUTER INTEGRATION IN THE CURRICULUM IN K-12 CENTRAL WISCONSIN SCHOOLS Gerald E. Gau

TU1-8
37 THE FUTURE OF EDUCATIONAL COMPUTING Lawrence Lipsitz

TU1-9
38 CRITICAL NON-TECHNICAL SKILLS FOR COMPUTER COORDINATORS AND ORGANIZING INSTRUCTIONAL TECHNOLOGY EFFORTS IN LARGER SCHOOL DISTRICTS Marilyn Gardner, Doreen Kelly

TU1-10
39 LIBRARY AUTOMATION AT UNION COLLEGE: THE BROADER PERSPECTIVE David V. Cossey, Jean K. Sheviak

TU1-11
40 TEACHING ETHICS IN COMPUTER SCIENCE Lillian N. Cassel, George G. Bear, John Beidler, Paul Gormley, Harriet Taylor, Richard Austing

TU2-2
41 EXPERT ADVISING SYSTEMS IN A COLLEGE/UNIVERSITY ENVIRONMENT Gerard Kiernan, Arnold Koltun, Edward N. Schwartz
46 WHAT HAPPENS IF YOU INTRODUCE AN INTELLIGENT TUTORING SYSTEM IN THE CLASSROOM: A CASE STUDY OF THE GEOMETRY TUTOR Yasmin Kafai

TU2-3
54 THREE DIFFERENT APPROACHES TO USING PROBLEMS WITH THE GEOMETRIC SUPPOSER Daniel Chazan, Richard Houde

TU2-4
58 ASSESSMENT FOR TEACHING MATHEMATICAL PROBLEM SOLVING: THE COMPUTER LINK Bethany N. Roditi, Judith Zorfass

TU2-5

59 RECOMMENDATIONS FOR UNDERGRADUATE PROGRAMS IN COMPUTER SCIENCE AND ENGINEERING A. Joe Turner, Allen B. Tucker, Keith Barker, Doris K. Lidtke

TU2-6

60 COMPUTER CONFERENCING FOR THE UNIVERSITY OF THE AIR STUDENTS IN JAPAN Hiroo Saga

65 A NEW GRADUATE TELECOMMUNICATIONS PROGRAM: THE UNIVERSITY AND CORPORATE WORLDS JOIN HANDS Stuart A. Varden

TU4-1

71 SPREADSHEETS AND CRITICAL THINKING Tony Jongejan

TU4-2

72 SOFTWARE DESIGN FOR LEARNING: ON LEARNING MATHEMATICS THROUGH INSTRUCTIONAL SOFTWARE DESIGN Idit Harel

87 LINGUISTICS, SCIENCE, AND MATHEMAITICS FOR PRE-COLLEGE STUDENTS: A COMPUTATIONAL MODELING APPROACH E. Paul Goldenberg

TU4-3

94 USING HYPERCARD IN A FIRST COLLEGE COURSE IN COMPUTING David E. Cummins, James R. Ullrich

97 FUNCTIONAL PROGRAMMING THROUGH EXPRESSION TEMPLATES: USING EXTENSIBILITY AND DYNAMIC SCOPE TO CUSTOMIZE CONTROL STRUCTURES IN LOGO Brian Harvey

TU4-4

104 THE APPLE CLASSROOM OF TOMORROW David Dwyer

TU4-5

105 LEARNING PROBABILITY CONCEPTS AND MODELING WITH COMPUTER SIMULATIONS Mike Perry

113 MATH IN OUR LIVES...WITH SPREADSHEETS Gertrude W. Abramson

TU4-8

117 STATE OF THE ART: AN EDUCATIONAL SOFTWARE DESIGNERS' FORUM Dave Grady, Walter Koetke, Lynne Anderson-Inman, Midian Kurland, Mary Cron, Mitch Resnick, Susan Schilling, Tim Barclay, Michael Hillinger

TU4-9

118 P.S. 2000: AN ACTION PLAN FOR USING TECHNOLOGY TO EDUCATE STUDENTS FOR THE 21st CENTURY Marilyn Gardner

THE COMPUTER INTEGRATION PROJECT IN THE ARLINGTON, VA PUBLIC SCHOOLS Sheryl Asen

IMPLEMENTATION OF COMPUTERS IN CLASSROOMS: THE PLAN AND THE PROGRESS Hilary Cowan, Judy Clifford, Linda Lindroth

TU4-10

120 AWARD-WINNING EDUCATIONAL SOFTWARE AND CURRICULUM INNOVATIONS IN HIGHER EDUCATION Robert Kozma, Jerome Johnston

TU4-11

121 CASE STUDIES OF COMPUTER USE: SPECIAL EDUCATION STUDENTS IN MAINSTREAM SETTINGS Judith Zorfass

TU5-2

122 EARLY CHILDHOOD INITIATIVE Marilyn Quinsaat, D. Midian Kurland

BRIDGING THE LEARNING GAP—EARLY PRIMARY COMPUTERS AND MANIPULATIVES Nancy Corning, Jack Halapin

TU5-3

123 HOW COMPUTER NETWORKS TRANSFORM LEARNING ENVIRONMENTS Reed A. Augliere

PROJECT SHINE Katherine Flannery Moss

KIDS' NETWORK James R. Samson

TU5-4

125 FACILITATING TECHNOLOGY-RELATED RESEARCH IN A SCHOOL SETTING Vivian Johnson, Greg Sales, Randall E. Johnson

TU5-5
126 TECHNOLOGY INSERVICE FOR K-10 TEACHERS OF MATHEMATICS George W. Bright
CAL POLY, POMONA'S COMPUTER LITERACY PROGRAM FOR CALIFORNIA TEACHERS Judith E. Jacobs, George Shail
THE BEGINNING TEACHER COMPUTER NETWORK, HARVARD GRADUATE SCHOOL OF EDUCATION Diane E. Beals

TU5-6
See End of Table of Contents

TU6-1
127 CATEGORIZATION AND DATA MODELING WITH CLASSIFICATION MACHINE Esther Inbal, Oded Susskind
THINKINGLAND: AN INTENTIONAL LEARNING PROJECT—THE DATABASE AND THE CULTURE Peter Skillen
DEVELOPING CRITICAL THINKING SKILLS: A COMPUTER-ASSISTED INTERVENTION PROGRAM FOR AT-RISK STUDENTS Kim MacGregor, Judi Repman

TU6-2
128 RESEARCH THEMES IN ARTIFICIAL INTELLIGENCE AND EDUCATION Chris Dede, Andy Molnar, Joe Psotka

TU6-3
129 TOUCH TYPING IN ELEMENTARY SCHOOLS—WHY BOTHER? Pamela Freyd, Jessica Kahn
135 THE EFFECTS OF COMPUTER ASSISTED INTERACTIVE FICTION ON THE DEVELOPMENT OF READING COMPREHENSION IN GRADE FIVE STUDENTS Robert F. Whitelock

TU6-4
140 TOOLS FOR MESSY MATHEMATICS: USED NUMBERS AND ELASTIC Andee Rubin, Susan Jo Russell

TU6-5
141 HYPERCARD: A TOLL FOR THE MATHEMATICS TEACHER Barbara Dubitsky, Rosemarie Buzzeo
HYPERCARD IDEAS FOR THE K-12 CURRICULUM Michael Land, Sandra Turner
HYPERTEXT CAI: A SUPPLEMENT TO AN ELEMENTARY SCHOOL BASAL READING PROGRAM Thomas C. Lovitt, Randall Boone, Kyle Higgins

TU6-7
143 ETHICS IN THE INTRODUCTORY COMPUTING COURSE Ronald E. Anderson, Steven L. Mandell, Daniel D. McCracken, Claire McInerney, Judith A. Perrolle

TU6-8
144 COMPUTERS: A LITERACY CONNECTION FOR YOUNG CHILDREN AT-RISK Trisha Ainsa
EFFECTIVE TECHNOLOGIES WITH YOUNG OR HANDICAPPED LEARNERS: TOUCHSCREENS AND THE MICROCOMPUTER Pamela Campbell
ENRICHMENT VS. REMEDIATION FOR AT-RISK STUDENTS Antonia Stone, Tom Harris, Arthur Wood

TU6-9
146 INSERVICE TRAINING: A COOPERATIVE APPROACH Sylvia Charp, Dave Brittain, Sandra Hogan, Kenneth Rossano, Frank Withrow

TU6-10
147 TEACHERS TEACH AS THEY ARE TAUGHT: TEACHING LOGO TO TEACHERS Nancy Cunniff
154 THE ELEMENTARY MATH AND TECHNOLOGY PROJECT: A MODEL FOR TEACHER ENHANCEMENT Ann Grady
159 TEACHERS AS TRAINERS: A STAFF DEVELOPMENT PLAN FOR SUCCESSFUL COMPUTER USE Vicki E. Baer

TU6-11
163 COMPUTING SKILL WORKSHOP: COMPUTER LITERACY IN THE CORE CURRICULUM AT CARNEGIE MELLON UNIVERSITY Carol Scheftic, Tracy Futhey
175 SUBJECT SEARCHING IN A UNIVERSITY ON-LINE CATALOG: IMPLICATIONS FOR SCHOOL LIBRARIES Sally E. Doyen, Daniel D. Wheeler

W1-1

180 RETHINKING LOGO PEDAGOGY USING A PROCESS APPROACH Daniel H. Watt, Molly L. Watt
187 MATHEMATICS EDUCATION: LEARNING FROM THE PROCESS APPROACH TO WRITING Rebecca Corwin, May C. Reinhart

W1-2

194 TEACHING INTRODUCTORY PROGRAMMING METHODS USING STRUCTURE EDITING: SOME EMPIRICAL RESULTS Dennis R. Goldenson
204 OBJECT-ORIENTED PROGRAMMING IN EDUCATION Mark Guzdial

W1-3

209 USING LOGO TO ENRICH THE ELEMENTARY MATHEMATICS CURRICULUM Bill Craig
BUILDING EXPERT SYSTEMS IN THE CLASSROOM: A PROBLEM SOLVING PROJECT Carolyn Knox-Quinn

W1-5

210 VIRTUAL REALITIES: FROM THE CONCRETE TO THE BARELY IMAGINABLE Stephen Marcus

W1-6

211 INTEGRATING COMPUTERS INTO THE STAR SCHOOLS PROJECT Thomas H. Saterfiel, Pamela S. Pease, Kenneth H. McKinley, Gail Arnal, Cecelia Lenk

W1-7

212 WHAT CAN YOU DO WITH A MINI-LAB? Robertta H. Barba
214 EDUCATIONAL SPECIFICATIONS FOR A COMPUTER RESOURCE CENTER Judy S. MacDonald

W1-8

220 EDUCATIONAL COMPUTING IN THE USSR Barbara Kurshan, Rachelle Heller, Karen Billings, Peter Copen, Lynne Schrum

W1-10

221 THE REQUIRED COMPUTER COURSE FOR EDUCATION MAJORS: A NATIONAL PERSPECTIVE Pamela Moore, Joyce S. Friske, Mildred D. Lintner, Bonnie Mathies, Charles Mlynarczyk, Lajeane Thomas, James Wiebe

W1-11

222 REAL TIME COMPOSITION: TEACHING WRITING WITH A LOCAL AREA NETWORK George H. Brett II, Trent Batson, Robert Cannon

W2-2

223 ACOT ROUNDTABLE DISCUSSION David Dwyer

W2-4

224 THE EFFECT OF COMPUTER USE AND STUDENT WORK STYLE ON DATABASE ANALYSIS ACTIVITIES IN THE SOCIAL STUDIES Don Rawitsch
MEDIATED INSTRUCTION AND LOGO: EXAMPLES AND EFFECTS Victor R. Delclos
RESEARCH DESIGN FOR EVALUATION OF MODEL APPLICATIONS OF COMPUTER TECHNOLOGY IN INDIANA CLASSROOMS Beverly Farr, Chris Franklin

W2-5

226 CURRICULUM INTEGRATION WITH COMPUTERS IN NEW ZEALAND SECONDARY EDUCATION: PROFESSIONAL DEVELOPMENT SUPPORT FOR TEACHERS David H. McKinnon, Patrick C.J. Nolan, Robert R. Sharp
CURRICULUM INTEGRATION WITH COMPUTERS IN NEW ZEALAND SECONDARY EDUCATION: A RESEARCH AND DEVELOPMENT PROJECT Patrick C.J. Nolan, David H. McKinnon, Robert R. Sharp

W2-6

227 ENRICHMENT OPPORTUNITIES FOR AT-RISK YOUTH Antonia Stone, Carol Edwards, Lylburn Dowling, Seth Caiklin

W3-1

228 CAN COMPUTERS TEACH? LESSONS FROM 25 YEARS OF RESEARCH Jerome Johnston

W3-2
229 IT'S NOT TOO LATE TO TAKE A "SHOP CLASS": LEARNING SYNTHESIS SKILLS THROUGH PROGRAMMING Elliot Soloway

W3-4
230 LANGUAGE: THE NEXT GENERATION OF WRITING TOOLS D. Midian Kurland, Art Bardige

W3-5
231 PROJECT STEP (SPACE TECHNOLOGY ENVIRONMENT PROGRAM) Dennis Kurtz, Peter Connors
THE EFFECTS OF CALCULATORS AND COMPUTERS ON THE MATHEMATICS CURRICULUM Kathleen Maury
INTEGRATING TELECOMMUNICATIONS INTO MIDDLE SCHOOL SCIENCE Michael J. Rush, Peggy Buchanan, Mark Bloede

W3-6
232 PRACTICAL GUIDELINES: USING TELECOMMUNICATIONS IN THE CLASSROOM Lynne Schrum, Len Scrogan, Cecilia Lenk

W3-7
233 VIDEODISCS AND HYPERCARD IN ART EDUCATION Kathy Rehfield, Karen Warner
AESTHETIC ANALYSIS OF SELECTED WORKS OF VAN GOGH Harold McWhinnie
WHAT CAN BE SAID DEVELOPMENTALLY ABOUT YOUNG CHILDREN'S COMPUTER DRAWINGS? David Alexander

W3-8
234 MODELS OF POLITICAL EFFECTIVENESS AT STATE AND LOCAL LEVELS Bobby Goodson, Sheila Cory, LeRoy Finkel, Geoffrey Fletcher

W3-9
235 THE IMPACT OF INFORMATION TECHNOLOGY ON THE ROLE OF THE SECONDARY SCHOOL: AN AUSTRALIAN PERSPECTIVE Eric Williams, Charlie McKavanagh
240 CONSEQUENCES OF DECLINING ADMINISTRATIVE SUPPORT FOR ELEMENTARY SCHOOL INSTRUCTIONAL COMPUTING John F. Beaver

W3-10
245 INFORMAL COMPUTER LEARNING IN MUSEUMS Adeline Naiman, Inabeth Miller

W4-1
246 TEACHERS BECOMING RESEARCHERS: AN EXPLORATION IN DYNAMIC COLLABORATION Bridget Somekh

W4-2
248 THE IMPORTANCE OF TEACHERS IN CENTER FOR TECHNOLOGY IN EDUCATION RESEARCH Karen Sheingold, Jan Hawkins, Cornelia Bruner, Margaret Honey

W4-3
249 DEVELOPMENT OF 2+2 PROGRAMS IN COMPUTER INFORMATION SYSTEMS: AN EXAMPLE OF SECONDARY AND COMMUNITY COLLEGE PARTNERSHIP IN CURRICULUM ARTICULATION Ann Roskow

W4-4
250 THE IMPACT OF COMPUTERS ON CLASSROOMS Margaret Riel
256 DISTANT DIALOGS: TELECOMMUNICATIONS IN TEACHER TRAINING R. Ann Zinck

W4-5
261 C.P.R. (COMPUTER PROCESSED RESEARCH) Barry Jahn
ENHANCING THE SOCIAL STUDIES CURRICULUM: USE OF VIDEO, HYPERTEXT, AND TELECOMMUNICATIONS Marianne Teague, Gerald V. Teague, G. Edison Weslow

W4-6
262 INFORMATION TECHNOLOGY AND SCIENCE EDUCATION: THE 1988 AETS YEARBOOK James D. Ellis, Paul Horwitz, Cecilia Lenk

W4-7
263 ELECTRONIC STUDYING: HYPERCARD AND OTHER APPLICATIONS TO HELP STUDENTS STUDY "BETTER" NOT "HARDER" Lynne Anderson-Inman

W4-9
264 RESEARCH IN THE APPLICATION OF COMPUTERS IN EDUCATION: TRENDS AND ISSUES Betty Collis

W4-10
265 COMPUTERS AND LIMITED-ENGLISH PROFICIENT STUDENTS: TOOLS FOR SUPPORTING SCIENCE LEARNING IN MIDDLE AND HIGH SCHOOL Ann S. Rosebery, Beth Warren, Bertram C. Bruce

W4-11
267 APPROACHES TO PREPARING EDUCATIONAL SOFTWARE DESIGNERS Shelly Heller, Bengt Bengtsson, Janet Graeber, Dianne Martin, Teresa Rosegrant, Renate Schulz-Zander

W5-1
268 LEARNING STYLES OF COMPUTER USERS Danielle R. Berstein
273 LEARNING TO USE versus USING TO LEARN Patrick Casey, Nancy Cunniff, Robert P. Taylor

W5-2
282 PREVENTING PROGRAM PLAGIARISM: EDUCATIONAL APPROACHES Marguerite Summers, William B. Evans, James J. Fletcher, John T. Gorgone, Cindy Hachney

W5-3
283 PROJECT HEADLIGHT - "SCHOOL OF THE PRESENT" Marilyn Schaffer, Paula Hooper, Isaac Jackson, Leo Biehler, Joanne Ronkin

W5-4
284 SAMPLE COMPUTER-BASED MATH CLASSES USING DIRECT MATHEMATICS MANIPULATION ENVIRONMENTS James J. Kaput

W5-5
285 THE ROCKY MOUNTAIN APPLE C.O.R.E. CLASSROOM Nancy Bathke
ENLIST MICROS II: IMPLEMENTING MICROCOMPUTERS IN SCIENCE TEACHING James D. Ellis

W5-6
286 THE EVALUATION AND SELECTION OF PROJECTION DEVICES FOR THE MICROCOMPUTER Doris M. Carey

W5-7
287 ELECTRONIC MUSIC IN THE SCHOOLS Gordon F. Duckel

W5-8
288 COMPUTERS FOR MATH, SCIENCE, AND EQUITY Jo Sanders, Mary McGinnis

W5-9
289 HOW SCHOOL DISTRICTS IN NEW JERSEY USE COMPUTER TECHNOLOGY TO MONITOR STUDENT PROFICIENCY FOR ASSESSED SKILLS Julia A. Stapleton
CONFIRMATION BIAS & LOGO—ONGOING RESEARCH Gregg Brownell, Dieter Zirkler
USE OF AN INTEGRATED LEARNING SYSTEM LAB IN AN ELEMENTARY SCHOOL AND ITS EFFECT ON THE TEACHERS AND THE STUDENTS Catherine Alifrangis, Jan Funk

W5-10
290 USING COMPUTER SIMULATION TO TEACH ECONOMICS: THE CASE OF PEOPLE EXPRESS John D. Sterman

W5-11
291 THE COURSEWARE DEVELOPMENT PROCESS: THE PENN STATE STYLE Marilynne W. Mathias, Carol A. Dwyer, Barbara A. Polka

W6-1
292 IN PREPARATION FOR COMPUTER SCIENCE AND ENGINEERING AT COLLEGE Keith Barker, Harry Lewis, Carmine Cardello, Naomi Amos, Doug Willet

W6-2
293 THE DAEDALUS PROJECT AND PRODIGY: ELECTRONIC CONNECTIONS BETWEEN EDUCATION AND BUSINESS Judith Sweeney, Chauncy N. Rucker, Suzanne Biegel

W6-3
294 PRODUCING AND USING SPEECH-BASED SOFTWARE Lynne Anderson-Inman, Mary Cron, Bill Adler, Micheal Hillinger, Richard Olson

W6-4
295 A NETWORKED STATISTICS/COMPUTER SCIENCE COMPUTER CLASSROOM Richard A. Alo'
THE VIDEOGRAPH PROJECT Robert J. Beichner

W6-5
296 MULTIMEDIA DESIGN Robert Mohl, Margo Nanny

W6-6
297 TEACHER-LINK: AN ELECTRONIC CULTURE Judi Harris

W6-7
300 NEW HAMPSHIRE'S GOVERNOR'S INITIATIVE FOR EXCELLENCE IN EDUCATION: COMPUTERS FOR TEACHERS Michael P. Goldsmith, Arthur Hammon, Larry Bickford
NEW HAMPSHIRE VIDEODISC DEVELOPMENT PROJECT: THE GRANITE STATE SAMPLER
Cynthia A. Dunlap, Marybeth Stock

W6-8
301 PREPARING COMPUTER-USING EDUCATORS IN 1989 Margaret L. Niess
304 GIVING TEACHERS THE TIME TO TEACH: THE CLASSROOM MANAGEMENT SYSTEM IN THE PUBLIC SCHOOLS IN SOUTH CAROLINA Gary West

W6-9
312 BEYOND DRILL AND PRACTICE: USING MICROCOMPUTERS WITH LEARNING DISABLED STUDENTS Susan Jo Russell, Rebecca B. Corwin, Janice Mokros, Peggy M. Kapisovsky

W6-10
313 LEXNET—TELECOMMUNICATIONS FROM EVERY CLASSROOM Frank DiGiammarino, Beth Lowd, Bruce Mallory, Amy Wagner
TELECOMMUNICATIONS BETWEEN SCHOOLS UTILIZING AN INTERMEDIATE UNIT ELECTRONIC BULLETIN BOARD IN THE DEVELOPMENT OF NEWSPAPERING SKILLS AMONG GIFTED AND REGULAR SIXTH GRADE STUDENTS Donald F. Orcutt, Donald M. Weinberger
TELECOMMUNICATIONS AND SECOND LANGUAGE LEARNING: INTERNATIONAL CROSS-CULTURAL CONNECTIONS Dennis Sayers, George Migliarini

W6-11
314 GETTING THE MOST OUT OF LARGE PROGRAMMING PROJECTS James L. Wolfe
318 TEACHING MACINTOSH ASSEMBLY LANGUAGE TO COLLEGE FRESHMEN W. D. Maurer
324 PROGRAMMING EFFORT AND PROGRAM QUALITY: QUANTIFYING THE RELATIONSHIPS
Barry J. Donahue

TH1-1
332 INTERACTIVE COACHING FOR CURRICULUM-BASED INFORMATION TOOLS Charles S. White, Jos. J. Beishuizen, Beverly Hunter, Jeanne C. Meadows

TH1-2
333 COMPUTING IN CAREERS/CAREERS IN COMPUTING: A KIT FOR TEACHERS, COUNSELORS, AND VOLUNTEERS John Lawson, Doris Carey, Bill Baird

TH1-3
334 THE FANTASTIC JOURNEY PROJECT Michael R. Muir
INTEGRATING SHORT AI PROGRAMS IN LOGO AND BASIC INTO CONTENT AREAS
Susan N. Kirkpatrick, Barbara Biglan
MICROCODE SIMULATOR Ted Mims, Michael Jones

TH 1-4
335 NEW DIMENSIONS FOR DATA EXPLORATION IN SOCIAL STUDIES Bryant Patten, Ian Jukes

TH1-6
336 USING VIDEODISCS AT VARIOUS LEVELS Dean Zollman

TH1-7
337 PAPER LIKE INTERFACE FOR EDUCATIONAL APPLICATIONS Doris Chow, Joonki Kim
345 INTERACTIVE VIDEO IN EDUCATION: THE DATABASE APPROACH—A U.K. PERSPECTIVE Wendy Hall

TH1-8
349 THE ROLE OF TECHNOLOGY IN EDUCATIONAL REFORM Don Rawitsch, Bob Pearlman

TH1-9
350 COMPUTERS AS CARRIERS OF SOCIAL CHANGE: USSR, COSTA RICA, AND THE USA Seymour Papert, Marilyn Schaffer, Clotilde Fonseca de Pacheco, Alexey Semenov

TH1-10
351 IDEAS FOR ELEMENTARY PRESERVICE FACULTY TO USE A COMPUTER IN TEACHER TRAINING Gary G. Bitter, Ruthie Blankenbaker, Craig Blurton, Mary Hatfield, Nancy Butler Songer, Shelley B. Wepner

TH2-2
362 MENTAL MODELS AND THE DESIGN OF INTELLIGENT TUTORING SYSTEMS John Frederiksen, Barbara White, Brian Reiser

TH2-3
363 GEOMETRIC SUPPOSER URBAN NETWORK: A MULTI-CITY ELECTRONIC LINKING OF URBAN MATH TEACHERS Grace Kelemanik, Pat Potocny, Tom Radford

TH2-4
264 TEACHING ASTRONOMY WITH MICROCOMPUTERS Philip M. Sadler

TH2-5
365 STATE SUPPORT FOR INSTRUCTIONAL TECHNOLOGY Michael Neill, Geoffry Fletcher, Cheryl Lemke, Harold McDermott, Elsie Brumbach

TH2-6
366 "WHO OWNS AN IDEA?" SOFTWARE PIRACY, "FAIR USE," AND SOFTWARE DEVELOPMENT ON CAMPUS: FACULTY AS ENTREPRENEURS vs. FACULTY AS CONTRIBUTORS TO WORLD KNOWLEDGE Steven Gibert, Linda Garcia, Francis Fisher

TH3-1
367 ADVENTURE GAMES: CONNECTING THE CURRICULUM Cathie Sherwood

TH3-2
372 TEACHING COMPUTER SCIENCE COURSES WITHIN MATHEMATICS DEPARTMENTS Judith L. Gersting, David Ballew, Ben Haytock, James Stager

TH3-3
373 COMPUTER SCIENCE MATERIALS AND TEACHER PREPARATION PROJECT Larry E. Liden
AN UNDERGRADUATE METHODS COURSE IN COMPUTER SCIENCE J. Philip East
DYNAMIC VISUAL ORGANIZERS FOR INTRODUCING SORTING ALGORITHMS Thomas Morgan

TH3-4
374 LEGO/LOGO: EXPERIENCES IN THE CLASSROOM Stephen Ocko, Mitchel Resnick, Linda Darisse, Michael Iovine, Geraldine Kozberg

TH3-5
375 DEVELOPMENT OF A PRECALCULUS MATHEMATICS COURSE BASED ON COMPUTERS Alfred Bork, Fran Antenore, Stephen Franklin
FUNCTIONS, STATISTICS, AND TRIGONOMETRY WITH COMPUTERS: ISSUES IN IMPLEMENTATION Catherine M. Sarther, Sharon L. Senk
INTEGRATING COMPUTERS AND COOPERATIVE LEARNING INTO THE MATHEMATICS CLASSROOM Barbara Shelly, Patricia Tinto

TH3-6
376 MODELS AND SIMULATIONS: HOW ONE HIGH SCHOOL BRIDGES THE GAP FROM COMPUTER SCIENCE TO SCIENCE Michael Haney, Lola Piper

TH3-9
377 DACOTT 21/20: DISTRICT AND COMMUNITY OF TOMORROW TODAY Gary Peterson, Cyndy Everest-Bouch

TH3-10
378 A COMPARISON OF TEACHER AND STUDENT EVALUATIONS OF EDUCATIONAL SOFTWARE Peggy Healy Stearns
TEACHER WORKSTATIONS: A KEY TO THE WIDER IMPLEMENTATION OF TECHNOLOGY THROUGHOUT THE CURRICULUM Clifton S. Harris
THE CURRY TEACHING SIMULATION: A GROWING REALITY Harold R. Strang

TH3-11
379 AUTHORING TOOLS: PROMISE AND REALITY David Bantz, Carol Scheftic, Tom Kurtz, Otmar Foelsche

TH4-2
380 FUTURE LEARNING SYSTEMS Paul Evans, Alfred Bork, Alphonse Buccino, Glen Hoptman, Robbie McClintock

TH4-3
381 THE BUDDY PROJECT: A HOME-SCHOOL PARTNERSHIP Anita Griffith, Peggy Vogel Hill, Kristen Hein

TH4-5
382 USER INTERFACES FOR THE ELDERLY: A RESEARCH EXPERIENCE FOR UNDERGRADUATES Margaret Christensen
387 FACULTY-UNDERGRADUATE PARTNERSHIPS IN COMPUTER SCIENCE RESEARCH Marc J. Lipman, Robert L. Sedlmeyer

TH4-6
390 CONNECTING COLLEGIATE CURRICULUM COMPUTING William E. Grimsley, Jr.
THE DEVELOPMENT OF A STUDENT-BASED EDUCATIONAL SOFTWARE SELECTION AND EVALUATION CENTER IN HIGHER EDUCATION Robert A. Gray

TU5-6
391 MAKING LEARNING VISIBLE: A HI-TECH, LO-TECH, AND NO-TECH APPROACH Hillel Weintraub

TH1-5 THE CHALLENGES OF INTEGRATING TECHNOLOGY INTO THE CURRICULUM: TEN TRENDS FOR LONG-RANGE PLANNING Patricia Sturdivant

IN SEARCH OF GENERAL PLANNING SKILLS

Sharon M. Carver
University of Rochester

Mark L. Walker
The Ellis School

Abstract

The goal of our research is to advance students' general planning skills, particularly those useful for open-ended tasks such as writing programs and essays. In this presentation, we discuss the general nature of planning skills and the specific skills demonstrated by novices and experts in both programming and writing. Based on this analysis of the goals, heuristics, and sub-skills necessary for effective planning (and lacking among most middle school students), we have designed a programming curriculum that focuses explicitly on the acquisition of planning skills. We report the results of a study designed to test the effectiveness of our curriculum for fostering planning skills among seventh grade students. Think-aloud protocols of students doing both list-processing and essay writing tasks revealed that the curriculum was effective for students who had developed accurate representations of the standard structures in each domain but not for those who lacked this prerequisite knowledge.

The Search for General Skills

Two educational goals have been used to justify instruction in almost every subject in the school curriculum: transmission of substantive subject-matter knowledge and enhancement of general problem solving skills. However, in recent years, there has been a justifiable dissatisfaction with our schools' ability to achieve either of these goals (Chance, 1986; Sadler and Whimbey, 1980). One widely-advocated remedy is to enrich students' educational environment by introducing computer technology into the classroom. Computer programming courses, in particular, have been proposed as a means for improving the distressingly weak thinking skills that American students have demonstrated on recent national assessments (NAEP, 1981, 1983). Unfortunately, while the number of schools teaching programming courses continues to increase, so does the number of studies failing to document significant improvements in students' thinking skills as a result of these courses.

We contend that most of these failures can be attributed to a lack of explicit focus on general thinking skills in the programming curriculum and that the lack of focus can be attributed to a lack of precise specification of what general thinking skills involve. We have taken a three-phase information-processing approach to remedying these problems. First, we use cognitive analysis and modeling techniques to identify the goals, heuristics, and sub-skills involved in a particular thinking skill. Then we design programming curriculum that focuses directly on the component processes we have identified. Finally, we use think-aloud protocols to assess students' use of the component processes on both programming and non-programming (transfer) tasks. The success of this approach has already been demonstrated with respect to debugging skills; elementary and middle school children using a Logo programming curriculum based on a cognitive analysis learned effective debugging skills and transferred them to non-programming contexts (Klahr and Carver, 1988; Carver and Risinger, 1987; Carver and Klahr, 1986; Carver, 1986). The goal of our current research has been to use the same approach to facilitate students' acquisition of generalizable planning skills.

In this paper, we will define and briefly describe the cognitive skills involved in effective planning. We will then discuss two studies documenting the planning difficulties middle school students have when writing programs in two different contexts. Next, we will describe the programming curriculum we designed to focus on the planning skills students failed to demonstrate. Finally, we will report the results of a study designed to assess students' acquisition of planning skills from a programming course using our curriculum and their transfer of planning skills to essay writing, a domain in which similar planning difficulties have been documented (Scardamalia and Bereiter, 1984).

Specifying the Nature of Planning Skills

Despite the vast literature on planning, there is little agreement about its definition. Most researchers agree that "planning is an activity that goes on whenever an individual formulates a goal in advance of acting and then directs action in systematic pursuit of this goal" (Forbes and Greenberg, 1982), yet their descriptions of the component processes vary widely. In a recent survey of the literature on planning skills, Scholnick and Friedman (1987) suggest that planning must be considered as a set of skills whose application depends on the planning context. In this section, we summarize the framework they have developed for organizing the vast set of planning skills; in the remainder of the paper, we will present our research in terms of this framework.

Representation. The planner's understanding of the spatial and causal structure of the domain constrains definitions of problem situations, goals, and means to achieving them. The representation may be general or specific, but general descriptions must still be translated into procedures consistent with specific conditions. *Veridical* representations are faithful renditions of the actual problem space; whereas, *idiosyncratic* representations are biased because of limited experience.

Choosing a Goal. Assuming that the goal is not given, effective planning involves four processes for choosing goals, whether they are well-defined (e.g. solve this algebra problem) or ill-defined (e.g. draw a picture). The *detection* process involves basic pattern recognition in which planners match existing conditions to known procedures and goals. The *monitoring* process keeps track of goals and subgoals as they are considered so that the *evaluation* process can determine the relative importance and feasibility of alternate goals. Finally, the *selection* process chooses the best alternative based on the evaluation and a set of criteria for the type of goals the planner wants (e.g. low-cost, sure-fire, etc.).

Deciding to Plan. The decision to plan must be made explicitly. The planner must believe that planning will be helpful in the existing situation and that the chosen goal is within reach. Having such beliefs requires knowing what strategies are available and when and why they are useful. Finally, the planner must have enough self-control to delay the impulse to act so that planning may occur. Strategies that have become routine are exempt from this decision process; in fact, execution of such strategies is rarely considered planning.

Formulating a Plan. Once the decision to plan has been made, the planner must either select a ready-made plan from an existing repertoire or formulate a new one. When the planner's goals are well-defined, planning involves searching in a top-down manner for existing concepts and procedures that can be combined to reach the goal. Such anticipatory planning requires simulating possible plans prior to action. Alternatively, if the goals are ill-defined or the environment constantly changing, the planner may proceed more opportunistically, working in both top-down and bottom-up fashion while combining planning and action. Such opportunistic planning requires flexible attention strategies, a large working memory, and well-developed metacognition.

Executing and Monitoring a Plan. Once a plan has been formulated, the effective planner monitors its execution to ensure that the strategies are successfully accomplishing their purpose. The amount of monitoring a planner does depends on the type of plan being executed. Highly automatized plans are generally tailored to specific situations; they require little effort for execution and little guiding feedback so little monitoring is necessary. Existing plans that are either modifiable or general can be more flexibly applied but require more monitoring to detect erroneous application. Metacognitive planning (i.e. general problem solving heuristics) enables the generation of new plans but requires the highest degree of monitoring. In any case, when the monitoring process detects difficulties, it then directs debugging of the plan.

Learning from a Plan. Several types of learning are possible from planning attempts. Planners may learn a specific plan or may learn that planning is useful in particular situations. They may refine their representation of the situation or alter the way in which they choose goals. None of this learning is possible, however, without monitoring and reflecting upon the planning process.

These six aspects of planning skill provide a useful structure for our discussion of students' planning skills in programming contexts. As Pea (1982) suggests, programming constitutes "a rich and explicit context for examining dynamic planning processes." In order to capture such dynamic processing, all of our assessments rely heavily on the analysis of students' verbal protocols to expose knowledge, strategies, and intermediate stages of processing. The following section reports the findings from protocol analyses of students with two different levels of experience writing programs in two different contexts.

Characterizing Students' Existing Planning Skills

Study 1

Our primary assessments of students' planning skills involved novice programmers in a list-processing environment. Our goal was to document the planning skills that students bring to the programming task, in the absence of formal instruction in planning.

Method. For the purpose of this study, we implemented a Logo list-processing course at a private girls' school. All 34 of the sixth graders participated in the study.[1] Students attended approximately 30 programming classes in groups of 5 or 6 over a one semester time-span. Prior to this course, the students' exposure to computers, in general, and to programming, in particular, was minimal. All classes were taught by the first author in a guided-discovery manner; none emphasized planning strategies.

In order to specify precisely which planning skills individual students used for writing programs, individual assessments of programming skill were administered at two points during the semester by a research assistant in a room separate from the regular classroom. At each test time, each student was given one class period to write a program according to one of two isomorphic program descriptions[2] which were counterbalanced with test time. One program was an interactive quiz on facts about Pennsylvania, and the other was an interactive interview designed for teachers. Both descriptions included the output that the program should produce when given a variety of inputs. Figure 1 shows isomorphic portions of the two program descriptions.

Students were asked to think and read aloud throughout the testing; all sessions were videotaped for later transcription.[1] Also, students were permitted to use paper, pencil, and their course folder at all times.

---

[1] This study was primarily designed to assess students' acquisition and transfer of debugging skills. See Carver and Risinger (1987) and Klahr and Carver (1988) for details.

[1] By isomorphic, we mean that both of the programs could have been written using the same structure, i.e., the same flow of control.

Results. Students' reference to and use of effective planning components was minimal on both programming tests. In general, students began programming as soon as they were given the program description, often without even reading the complete description. They wrote code in its final form, in the order in which it should be executed. Subjects did not anticipate the need for subprocedures; rather they decided to subdivide their code only when faced with a complex conditional statement. Monitoring was almost non-existent, either in the form of program testing and debugging or in the form of mental execution.

Representation. For these novices, limited representations of the programming domain severely constrained their planning skill. The protocols revealed idiosyncratic representations with an incomplete understanding of the command structure and a shallow understanding of what actions each command instructs the Logo interpreter to perform. Thus, students have difficulty applying their knowledge to particular programming situations. Conditional statements (IF) pose particular problems in this regard because they require three inputs (the condition, the true case, and the false case), each of which may be composed of more than one Logo object.

Choosing a Goal. The program descriptions provide only ill-defined goals because they include no information about how the program should be structured. Students could still recognize patterns within the program description that could be matched to known procedures. At the lowest level, most students recognize that to get an interactive program to "say" something to the user, the PRINT command must be used. The protocols reveal similar rules about what command to use to get input from the user (a combination of MAKE and READWORD or READLIST) and what command to use to respond conditionally (IF). What is missing from the protocols, however, is a higher level plan that combines these elements or indicates appropriate subprocedure structure.

Deciding to Plan. Without adequate representations and higher level goals, students rarely decide to plan. They either lack appropriate strategies or the strategies they know are too demanding to execute in the testing situation. More importantly, their impulse to act without delay is too strong to overcome; most of them enjoy programming and shun "hands-off" time, so they begin coding almost immediately.

Formulating a Plan. Anticipatory plans are rare, especially because most students do not take the time to read the program description fully and, therefore, have not set well-defined goals. For example, they do not plan to divide their program into subprocedures. When they are faced with a complex conditional statement, however, they made decide opportunistically to create two subprocedures to deal with the two different outcomes of the condition.

Executing and Monitoring a Plan. Since students did not demonstrate a knowledge base of plans, they rarely used monitoring. They rarely referred to existing code when generating new code, even when the existing code was analogous or contained information, such as variable names, that was being reused. Even if the students had been executing plans, they may not have known how to judge their success because they have little knowledge about what makes a "good" program (e.g. concise code with appropriate use of subprocedures). For example, the students who did write subprocedures frequently failed to call the subprocedure so they created disconnected code. Students rarely discovered such errors because they rarely tested their code either interactively or by means of mental simulation.

Learning from a Plan. Since there was little planning, there was little learning about planning. Even though students often got frustrated when writing programs, they could not infer that planning would be helpful or what planning strategies to use. Students did show increased facility with basic commands on the second test, but these representations were not sufficiently veridical to support effective planning.

In addition to the formal planning assessments in Study 1, we collected intermediate and final versions of all of the students' in-class projects. For these projects, only the general nature of the project was given (e.g. write an interactive program in which the computer is trying to make friends with the user); no specific output descriptions were provided. In order to determine whether students perform similarly in class and during testing, we compared the first and final versions of two projects (one early in the term and one late in the term). For both projects, only about 30% of the subprocedures that exist in students' final versions were included in the first versions and less than 20% of the procedure calls had been coded. This result indicates that, similar to the testing results, students are not establishing the whole structure of the program when they begin coding and that they are writing subprocedures that are not connected to the main procedure in any way. We also examined the fate of the first-version code. 70% of it appeared in the final version in either identical form or with minimal changes (e.g. spelling correction). Only 30% had been elaborated or replaced. Thus, students are attempting to write finalized code on the first pass, as in the testing situation.

Study 2

In order to test the generality of our findings, we assessed the planning skills of students with extensive programming experience in a graphics environment.

---

[1] Space constraints prohibit discussion of transcription methods; sample protocols will be included in the presentation, however.

Method. These assessments took place in the context of an existing Logo graphics project in a well-integrated city public school. Prior to the assessments, students had extensive unstructured programming experience, approximately 200 hours over an 18-month period, under the direction of their regular teachers. Fifteen of the fifth grade students, who were selected by their teachers as the best programmers, participated in the study. Since the students' programming experience had been unstructured, we assume that there had been no particular emphasis on planning strategies.

In this study, one individual assessment of programming skill was administered by the first author and a graduate assistant in a room separate from the regular classroom. Each student was given one class period to write a program to draw a particular picture (Figure 2). All procedures were identical to Study 1 except that there had been no course materials.

Results. Despite 200 hours of programming experience, the students' effective planning was minimal.

Representation and Choosing a Goal. Most students demonstrated facility with the basic graphics commands, FD, BK, LT, RT, PU, and PD. Most did not, however, have veridical representations of the REPEAT command or the use of subprocedures, particularly for generating portions of the picture that were identical (e.g. the ears and/or stalks of corn). When probed about the REPEAT command, students' responses indicated that they knew one plan for using REPEAT to make a square and one for a circle. They were not aware, however, of how to alter the square plan to make a rectangle (for the silo and barn) or how to alter the circle plan to make an arc (for the ears of corn or the top of the silo). Therefore, they resorted to combining individual move and turn commands. Only one student mentioned, in advance, the possibility of using subprocedures to draw each piece of the picture, particularly the identical pieces of the corn. The others did not mention creating subgoals, and they generated spaghetti code (one long procedure with no meaningful line breaks).

Deciding to Plan. Two aspects of the students' behavior indicate decisions to plan, in the sense of assuring the success of their programs in advance. Prior to writing code to position the graphics cursor, several students entered the interactive mode, moved the cursor to the desired location, and used the POS command to get the coordinates of that position. These coordinates could then be used with the SETPOS command in the program. These students realized that such advanced planning would be useful and they had a well-developed strategy ready. Other students decided to test all of their code interactively before adding it to their programs.

Formulating a Plan. Despite the advanced preparation indicated above, the students' planning was still opportunistic. No students determined all of the necessary cursor coordinates before beginning to code; they determined each coordinate as it was needed. In addition, students generating code interactively still approached the picture in line-by-line fashion.

Executing and Monitoring a Plan. Monitoring of programming behavior, in general, was minimal. Students generated large amounts of code before testing it and then had difficulty locating the erroneous commands because their programs had little substructure. In addition, students using the interactive positioning strategy were highly practiced and highly reliable, so they needed to do little monitoring during execution. On the other hand, the interactive code generation is prone to frequent error, yet the students who used it did little monitoring.

Learning from a Plan. Once again, learning from plans is difficult when plans are not being used. In this study, however, the experimenters did intervene with instruction about the general use of the REPEAT command (since none of the students understood it and the picture we had chosen included so many arcs). We essentially gave students an arc plan, REPEAT a [FD b RT c], and explained the relationship between the inputs (a X c determines the angle of the arc and a X b the size). After such explanations, students often used the arc plan later in the programming session or in a subsequent debugging session (not described here).

Similar difficulties planning written essays

Scardamalia and Bereiter (1984) have identified planning as one of the major difficulties students have with writing. Their research provides a striking parallel to our descriptions of students' difficulties with writing programs.

When writing essays, students use a simple "knowledge-telling" strategy; they write everything they know about the given topic as it comes to mind. They have not represented the structure of various genres well-enough to identify which type to use. In essence, they cannot decide to plan because they do not have available plans or plan units and they have not learned that planning can be useful. The "knowledge-telling" strategy is essentially opportunistic, with opportunities considered as synonymous with ideas.

In addition, students rarely plan or revise their prose. They do not consider all of the information they want to present and then organize it according to a particular scheme. Furthermore, without such schemes or some criteria for assessing their writing, students' revisions are limited to spelling and low-level grammatical corrections. The structure of the essay is rarely altered. According to Flower and Hayes (1986), students rewrite rather than revise; if they determine that a sentence is inadequate, they write a new one instead of correcting the original one.

Similarity among experts writing programs and essays

The similarity between performance on program and essay writing tasks extends beyond the novice level to expert behavior. Littman et al. (1986) and Fisher

(1986) have found that the planning behavior of expert programmers depends heavily on building an initial representation of the problem, which involves decomposing it into goals and subgoals for which the programmer already knows or can readily generate a plan of action. In addition, experts view the programming process as cyclic; in other words, they know that their initial plan will be found insufficient and they plan to revise. They typically begin by coding a prototype version of the program and then work on improving its efficiency and interface. In addition to revising, they also decide to rewrite code when they discover an incorrect plan.

Flower and Hayes (1986) have identified similar planning strategies among expert writers. They begin by representing their writing assignment in terms of the goals that will guide and constrain their plans (e.g. the intended audience). They then decompose their knowledge about the topic into an organized hierarchy. As with programmers, they expect that their first draft will not be perfect; they consider revision a major part of the writing process. They focus on improving the whole text with respect to their representation and goals, and they use both revision and rewriting.

The goal of our current line of research is to advance students' general planning skills, particularly those for open-ended tasks such as writing programs and essays. We have used the specification of planning skills described earlier and the literature describing expert planning strategies to design a programming curriculum that explicitly focuses on planning skills. We then assessed students' use of planning skills on both program and essay writing tasks to determine where advances in planning skills occurred as a result of our intervention.

Designing Programming Curriculum to Focus on Planning Skills

Our programming curriculum was designed for use by students who had already mastered the basic Logo list-processing commands and the basic flow of control principles, including the use of conditional statements and subprocedures. By requiring this prerequisite knowledge, we targeted students whose representations of the programming domain were growing beyond idiosyncratic to more veridical ones. In addition, we could begin focusing on planning skills at the beginning of the course while reviewing concepts that were already familiar (to avoid imposing unnecessarily high processing demands by introducing new programming concepts and new planning strategies simultaneously).

We developed a 5-phase planning sequence including:

Program Planning. The first phase involves specification of what should happen when a user runs the program as well as the content and form of the information exchanged between Logo and the user. Following this specification, program actions must be grouped into procedures based on a small set of standard rules (e.g. "separate actions that need to be repeated from those that don't").

Procedure Planning. The second phase involves further planning of each of the procedures identified in the first phase. At this point, the goals for each procedure must be mapped to programming constructs that can be used to accomplish them. No actual code is written, but commands can be selected and procedure and variable names specified.

Drawing a Flow Diagram. The third phase involves specification of the relationships between procedures so that the organization of the whole program is clear. Opportunities for creating reusable subprocedures may become apparent at this point if they have been missed before.

Writing the Program Skeleton. Coding does not begin until the fourth phase, which involves following a few simple guidelines for coding the "bare bones" of the entire program before "fleshing it out." Students are encouraged to code a simplified version of each procedure so that they can establish the correct flow of control. This simplified version includes only as much code as is necessary for the program to run. For example, PRINT [instructions] can be used in place of a series of PRINT statements that actually give complete instructions.

Finalizing the Program. The fifth phase involves adding the finishing touches to make the program robust (e.g. by including error checking), user friendly (e.g. by including clear instructions), etc. At this point, students are also encouraged to seek peer criticism in order to improve their projects.[1]

To encourage students to monitor their plans, we frequently emphasized the cyclic nature of planning. For example, one of the first handouts listing the five phases included the following caveat:

> When you first think of a project idea, it's best to start by following these five stages in order. But, you'll soon realize that as you begin to work, you'll have new ideas about what to include in your project and you'll encounter problems that you had not anticipated...In other words, you'll be more likely to **cycle through these 5 stages several times** than to complete each stage before moving on to the others.

The planning sequence was introduced in the context of simple examples at the beginning of the course. The teachers, both the first and second authors, provided scaffolding by doing parts of the work so that students did not have to execute all of the planning phases at once (See Brown and Palincsar, in press). Gradually, the students accepted more and more responsibility for the planning. Then the 5-phase sequence was discussed explicitly and practiced on larger-scale projects. Throughout the remainder of the 30 hour course, students were required to use the planning sequence on all course projects, each of which involved more advanced programming concepts.

---

[1] Due to space limitations, the entire curriculum cannot be included in the paper. Copies will be available at the conference.

Assessing Students' Acquisition of Planning Skills

Study 3

The goal of the third study was to test the effectiveness of our curriculum.

Method. Sixteen of the subjects from Study 1 participated in Study 3 during the second semester of their seventh grade year in the context of a second 30-hour list-processing course using the curriculum described above.

Students' planning skills were assessed individually at three times during the course, immediately after the introduction of planning strategies, midway through the course, and near the end of the course. At each test time, students were given one of two isomorphic versions of either a programming or a writing task. Since only three rounds of testing were possible, each student took two programming tests and one writing test or vice versa. The two programming tasks (a dating service and a talent search) included many opportunities for planned structure, such as repeated actions, random components, and complex conditional responses. The writing tasks requested defense of an opinion based on pros and cons from a variety of perspectives. They varied in terms of the structure of the supplemental information students were given about the topic; students were given either no information, a disorganized list of information, or a structured table of information. Figure 3 shows one isomorph of the writing task and the alternate types of information students were given (the other isomorph involved an essay about whether uniforms should be required).

To gather evidence for the use of the five phases included in the planning instruction, we asked students to think and read aloud throughout the testing; all sessions were videotaped for later transcription. For programming tests, students were permitted to use paper, pencil, and their course folder. For writing tests, students had access to paper, pencils, a dictionary, and a thesaurus.

Programming Test Results. The most striking result of this study was the vast individual differences in the ability and the need to use the planning strategies taught in class. About half of the students still had difficulty representing the basic programming commands and demonstrated little knowledge of standard programming plans. They approached the task linearly, did not attempt to understand the overall organization of the output, rarely used subprocedures, generated final code on the first pass, and rarely tested their programs. They either did not learn the planning strategies or were not able to implement them because of inadequate representations.

Almost all of the other students demonstrated veridical representations of the basic programming commands and of elementary programming plans, such as the typical PRINT, MAKE, IF sequence for getting and responding to input, using subprocedures for reusable code, etc. They determined the organization of the output and established their subprocedure boundaries in advance, either by drawing boxes on the output sheet or by listing and grouping the program's actions on paper. They also chose procedure and variable names in advance to assure consistency. None drew flow diagrams, though several mentioned that doing so would help with debugging later. These students then began to code in skeleton form. They also tested and debugged their code frequently.

Two types of planning remained difficult even for these students: *invisible actions* and *merged plans*. Invisible actions are those that have no immediate, observable effect on the program output. For example, in order to vary the responses given at a particular time, the student must code a statement to choose a random response before coding the PRINT statement to make the response. The first of these actions is *invisible* on the output sheet and was often omitted by students. Merged plans involve inserting one standard plan into another, such as making a random choice or printing a variable value within a standard PRINT statement. Students often had difficulty dealing with such merger.

There was one student for whom the level of programming required by the tests had become so automatic that planning was, for her, unnecessary. She used a standard structure for all of her programs, generally calling subprocedures for each PRINT, MAKE, IF sequence or related groups of sequences. She made excellent use of the output sheet to determine which code would be reuseable and which responses needed to be chosen at random.

For all of the students who took two programming tests, the performance on both tests was quite similar. Those whose representations of the programming task were insufficient to support planning on the first test were unlikely to improve enough to support planning on the second test and those who were able to plan well on the first test continued to do so on the second test. In other words, the instruction was effective only when the students had already developed effective representations of the programming domain. In these cases, the effect was immediate and dramatic. Students began using the strategies early in the course, their programs improved, they worked more quickly (because they made fewer errors), and they experienced less frustration than they had in the previous course.

Essay Writing Test Results. On the writing tests, students demonstrated similar individual differences that were consistent across tests. Many students used the "knowledge-telling" strategy described earlier. They paid little attention to the supplemental information provided or did not organize it further. Generally, they wrote one long paragraph containing a list of points without organizing them according to pros and cons or the type of perspective. They revised only for clarity and spelling.

Other students, however, considered the additional information carefully, adding structure where it was lacking. One student, who received no supplemental information, generated a list of pros and cons before beginning to write. These students generally followed standard plans for paragraphs (topic sentence, supporting evidence) and essays (introduction, body, conclusion) and often wrote outlines before beginning to write the essay. For these students, revision went beyond low-level corrections to include adding examples, making transitions, and adding concluding sentences.

In terms of specific evidence for transfer from the programming instruction, it was not always the case that students who were able to use planning skills in the programming domain were the ones who used them for writing essays. In both domains, the use of planning skills is constrained by existing representations, so transfer can only occur if the appropriate representations already exist for writing essays. In one case, however, a student did mention the idea of writing a "skeleton" essay before the final product.

Future Directions in the Search for General Planning Skills

We have provided tentative evidence for the effectiveness of our planning skill curriculum for programming courses; however, further attempts to evaluate it would be premature since many students apparently lack the necessary prerequisite representations of the programming domain. Our next steps, therefore, will be to design additional curriculum to foster the development of veridical representations beginning with individual commands and leading to larger programming plans, particularly those that are useful for getting and responding to input. Students also need further instruction in determining appropriate uses for subprocedures. We will also pay particular attention to *invisible actions* and *merged plans* since almost all of the students found them difficult.

In terms of teaching generalizable skills, those that will be useful across a wide variety of domains, our research highlights the dependency of general skills on subject-matter knowledge. Our information-processing approach to improving students' general problem solving skills can only be effective when students already have, or we also provide, the relevant prerequisite knowledge in each of the domains where we expect students to apply the general skills.

References

Brown, A.L. and Palincsar, A.S. (in press). Reciprocal teaching of comprehension strategies: A natural history of one program for enhancing learning. In J.B. Borkowski and J.D. Day (Eds.) *Intelligence and cognition in special children: Comparative studies of giftedness, mental retardation, and learning disabilities*. Norwood, NJ: Ablex.

Carver, S.M. (1986). *Logo debugging skills: Analysis, instruction, and assessment*. Doctoral Dissertation, Department of Psychology, Carnegie Mellon University.

Carver, S.M. and Klahr, D. (1986). Assessing children's debugging skills with a formal model. *Journal of Educational Computing Research*, 2(4), 487-525.

Carver, S.M. and Risinger, S.C. (1987). Improving children's debugging skills. In G. Olson, S. Sheppard, and E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop*. Norwood, NJ: Ablex.

Chance, P. (1986). *Thinking in the classroom*. New York, NY: Teachers College Press.

Fisher, C. (1986). *How do programmers program: Coping with complexity*. Master's Thesis, Department of Psychology, Carnegie Mellon University.

Flower, L. and Hayes, J.R. (1986). Detection, diagnosis, and the strategies of revision. *College Composition and Communication*, 37(1), 16-55.

Forbes, D.L. and Greenberg, M.T. (1982) Children's planning strategies. *New Directions for Child Development #18*. San Francisco, CA: Jossey-Bass.

Klahr, D. and Carver, S.M. (1988). Cognitive objectives in a Logo debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology*, **20**, 362-404.

Littman, D.C., Pinto, J., Letovsky, S., and Soloway, E. (1986). Mental models and software maintenance. In E. Soloway and S. Iyengar (Eds.), *Empirical Studies of Programmers*. Norwood, NJ: Ablex.

National Assessment of Educational Progress (1981). *Reading, thinking, and writing: The results from the 1979-1980 national assessment of reading and literature* (11-L-01). Denver, CO: Educational Commission of States.

National Assessment of Educational Progress (1983). *The third national mathematics assessment: Results, trends, and issues* (13-MA-01). Denver, CO: Educational Commission of States.

Pea, R.D. (1982). What is planning development the development of? In Forbes, D.L. and Greenberg, M.T. Children's planning strategies. *New*

Directions for Child Development #18. San Francisco, CA: Jossey-Bass.

Sadler, W.A. and Whimbey, A. (1980). Teaching cognitive skills: An objective for higher education. *National Forum*, Fall, 43-46.

Scardamalia, M. and Bereiter, C. (1984). Teachability of reflective processes in written composition. *Cognitive Science*, 8, 173-190.

Scholnick, E.K. and Friedman, S.L. (1987). The planning construct in the psychological literature. In S.L. Friedman, E.K. Scholnick, and R.R. Cocking (Eds.), *Blueprints for Thinking: The Role of Planning in Cognitive Development*. New York, NY: Cambridge University Press.

```
Portion of Output A                          Isomorphic Portion of Output B

Do you like living in Pennsylvania?          Are you married?
NO                                           NO
Too bad, do you like working here?           Do you have a boyfriend/girlfriend?
YES                                          YES
What is your job?                            What is his/her name?
TV REPAIR                                    DAVE
TV repair is an interesting job.             Dave is a nice name.
```

Figure 1: Partial Output for the Isomorphic Programming Tests in Study 1.

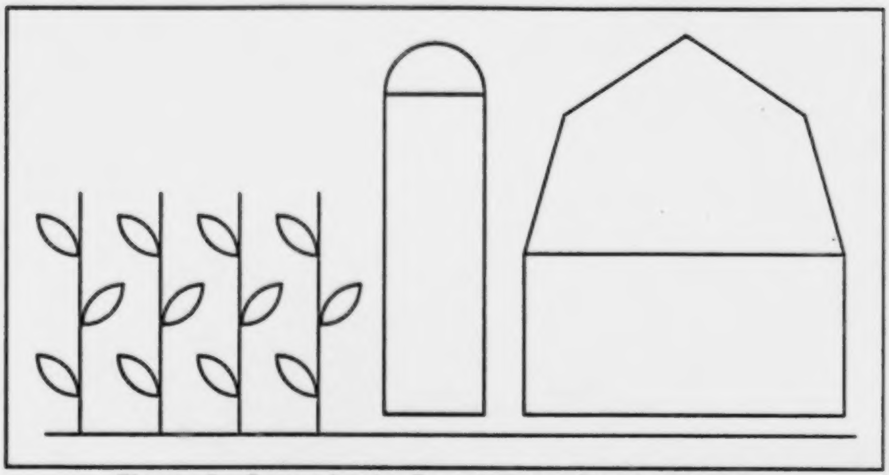


Figure 2: Output for the Programming Test in Study 2.

Standard Writing Task

Think about whether students should be given grades. Write about the positive and negative aspects of grades, both from your point of view and from other people's points of view (such as teachers, parents, and other students). Also, include enough detail so that the person who reads your paper will understand this important issue and why he or she should agree with your position on it.

Sample of Disorganized Survey Information

Mr. Daily (English Teacher) argued that grades create more work.
Lisa McCabe (Grade 7) likes knowing where to try harder.
Mrs. McCabe (Lisa's mother) said that grades help parents to know how their children are doing.
Greta Kuhn (Grade 11) said she hates the stress grades cause.

Sample of Structured Survey Information

	Pros	Cons
Parents	Know how kids are doing Know where kids need help	More to worry about
Teachers	Know how kids are doing	More work

Figure 3: Writing Test and Sample Survey Information in Study 3.

COMPUTER PROGRAMMING AND PROBLEM SOLVING ABILITIES: A META-ANALYSIS

Yuen-Kuang Liao and George W. Bright
University of Houston

ABSTRACT

Although claims regarding the cognitive benefits of computer programming have been made, results from existing empirical studies are conflicting. To make a more reliable conclusion on this issue, the meta-analysis research method was used to synthesize existing research concerning the effects of computer programming on problem-solving abilities. Nineteen studies were located from three sources, and their quantitative data were transformed into a common scale -- Effect Size. The results of the analysis show that 16 or 84% of the study-weighted effect sizes were positive and favored the computer programming group. The overall grand mean of the study-weighted effect size for all 211 comparisons was 0.41. This suggests that students having computer programming experiences scored about 16 percentile points higher on various cognitive-ability tests than students who did not have programming experiences. However, none of the nine classification variables selected for this study (e.g., year of publication) had a statistically significant impact on the mean study-weighted effect size. The findings suggest that the outcomes of learning a computer language are beyond the content of that specific computer language. The results of this study also provide teachers a mildly effective approach for teaching students problem-solving skills in a classroom setting.

The teaching and learning of programming has quite rapidly become an important topic among educators. The widespread availability of microcomputers has led to their rapidly increasing use in the nation's schools, and a huge number of these schools, in particular at the elementary and secondary levels, are using computers for instruction in programming. According to Pea and Kurland (1984a), several million precollege US students receive instruction on computer programming each year.

One reason frequently cited by researchers and decision-makers in education for placing such a heavy emphasis on programming is its presumed impact on problem solving abilities beyond programming activities (e.g., Coburn, Kelman, Roberts, Snyder, Watt, & Weiner, 1982). Nickerson (1982), for example, maintained that

> Perhaps the basic reason for the belief that programming might be an effective vehicle for the acquisition of generally useful cognitive skills is the assumption that programming is prototypical of many cognitively demanding tasks. It is a creative endeavor requiring planning, precision in the using of language, the generation and testing of hypotheses, the ability to identify action sequences that will realize specified objectives, careful attention to detail, and a variety of other skills that seem to reflect what thinking is all about. (p. 45)

Observations of expert adult programmers also indicate that programmers explicitly employ important problem solving strategies, such as decomposing problems into modules, using analogical reasoning, and systematically planing, coding, and debugging their programs (Nickerson, 1982). These complex cognitive skills may be transferable to other situations beyond programming activities, but there is no automatic guarantee of this.

There are several strong claims being made about the positive benefits in cognitive skills of teaching computer programming to children. Papert (1980), one of the creators of Logo and a leading exponent of the use of computer programming to expand children's intellectual power, recommends teaching students how to program. He believes that programming allows children to create their own learning environment. The most important claim he makes is that learning Logo enhances problem solving skills by providing concrete experiences that promote thinking at a formal operational level. Formal operational thinking is described by Piaget (1952) as the ability to construct relationships, make inferences, and hypothesize. It has also been claimed that Logo can teach mathematics concepts, alleviate a fear of math, provide an environment in which children feel in control and can succeed, positively affect attitudes toward learning, and in general teach children how to learn (Papert, 1980; Watt, 1982). Lochhead and Clement (1979) also proposed that the computer programming environment holds the promise of being an effective device for cognitive process instruction -- teaching how, rather than what, to think.

Although claims concerning the cognitive benefits of computer programming have been made, research results are conflicting. Existing empirical studies conducted by Clements (1986, 1987), Rieber (1986), Clements and Gullo (1984), Ehrlich, Abbott, Salter, and Soloway (1984), and Statz (1974) all report significant gains in various aspects of problem solving abilities following programming experiences. Most of these studies investigated the effects of the Logo programming language on problem solving abilities for students at the elementary school level. On the other side, studies conducted by Dalton (1986), Rucinski (1986), Kurland, Pea, Clement, and Mawby (1986), Horner and Maddux (1985), Pea and Kurland (1984b), and Cheshire (1981) have indicated that there is little, if any, transfer of learning from the programming situation to similar non-programming tasks. These studies varied in instructional levels (kindergarten through college) and instructional languages (Logo, BASIC, and some mixtures of programming languages).

Several syntheses of research on computer programming and problem solving ability have been done (i.e., Blume,1984; Clements, 1985; Johanson, 1988). However, none of them used the meta-analytic approach. Owing to the contradictory evidence provided by existing research in the area, the results from this meta-analysis will help provide clearer conclusions.

Procedure

The research method used in this study is the meta-analytic approach which was similar to that described by Glass, McGaw, & Smith (1981). Their approach requires a reviewer (a) to locate studies of an issue through objective and replicable searches; (b) to code the studies for salient features; (c) to describe studies outcomes on a common scale; and (d) to use statistical methods to relate study features to outcomes (Kulik, Kulik, & Bangert-Drowns, 1985).

The purpose of this study is to synthesize and analyze the research studies on effects of two instructional approaches. It is important to define these approaches of instruction to be used to provide for proper selection of appropriate studies clarification results.

computer programming -- using the concepts of program, sequence of instructions, variables, recursion, etc., to write solutions to a set of problems dealing with string processing operations

computer programming instruction -- classes using computer languages, such as Logo, BASIC, Pascal as an instructional tool to teach students.

non-computer-programming instruction -- classes not using any computer programming as an instructional tool to teach students.

Data Sources

The studies considered for use in this meta-analysis came from three major sources and were published from 1973 to the present. One large group of studies were generated primarily through computer searches of Education Resources Information Center (ERIC), which consist of two files, Research in Education and Current Index to Journals in Education. A second group of studies came from Comprehensive Dissertation Abstracts. A third group of studies was retrieved by branching from bibliographies in the documents located through review and computer searches.

Several criteria were established for inclusion in the present study.

1. The studies had to assess the relationship between computer programming and cognitive skills related to problem solving abilities such as planning skills, thinking skills, reasoning skills, and metacognitive skills.
2. Studies had to take place in actual classrooms. These was no restriction on grade level.
3. The studies had to provide quantitative results from both a computer-programming and non-computer-programming classes.
4. The studies had to be retrievable from university or college libraries by interlibrary loan or from the Education Resources Information Center, or Dissertation Abstracts International, or University Microfiche International.

There are also several criteria for eliminating studies or reports cited by other reviews: (a) the studies did not report sufficient quantitative data in order to estimate Effect Sizes; (b) the studies reported only correlation coefficients -- r value or chi-square value; (c) the studies could not be obtained through interlibrary loans or from standard clearinghouses. Nineteen studies were located that fit the criteria, seven studies from ERIC, seven studies from journals, and five dissertations.

Outcome Measures

The instructional outcome measured most often in the 19 studies was student learning, as indicated on standard cognitive-skill tests at the end of the program of instruction. For statistical analysis, outcomes from a variety of different studies with a variety of different instruments had to be expressed on a common scale. The transformation used for this purpose was the one recommended by Glass et al.(1981). To reduce measurements to a common scale, Glass and his colleagues code each outcome as an Effect Size (ES), defined as the difference between the mean scores of two groups divided by the standard deviation of the control group. For those studies that did not report means and standard deviations, F or t value was used for estimating ES.

In most cases, the application of the formula given by Glass and his colleagues is quite straightforward. But in some cases, when more than one value is available for use in the formula of ES, the value which measures outcomes most correctly was selected. For example, some studies reported both differences on posttest measures and differences in pre-post gains, and some studies reported both raw-score differences between groups and covariance-adjusted differences between groups. In such cases, pre-post gains and covariance-adjusted differences were selected for estimating ES.

In addition, when studies used more than one type of control group (e.g., programming vs. CAI; programming vs. non-programming; and CAI vs. non-programming), only the comparison of programming and non-programming was retained. In other cases, several subscales and subgroups were used in measuring a single outcome (e.g., those that reported separate data by gender or grade). In such cases, each comparison was weighted in inverse proportion to the number of comparisons within the study (i.e., 1/n, where n = number of comparisons in the study) so that the overweighting of ES of a study could be avoided (see, for example, Waxman, Wang, Anderson, & Walberg, 1985, p. 230).

Variables Studied

Variables chosen for this study include type of publication, year of publication, instructional level, language studied, number of comparisons, sample size, type of design, duration of treatment, and statistics. The nine categorical variables were employed as factors in an analysis of variance (ANOVA) to investigate whether there were significant differences between variables and the effect size. Research descriptions for most of these variables are shown in Table 1.

Coder reliability

To obtain more reliable outcomes from coding, one investigator coded each of the studies on each of the variables. As a check for accuracy, a second investigator coded four studies independently including two from journals, one from ERIC, and one from Comprehensive Dissertation Abstracts. The intercoder agreement for these four studies exceeded the 85% criterion.

Results

Table 2 reports the number of comparisons, and the study-weighted effects sizes from the 19 studies included in the present study. The overall grand mean of the study-weighted effect sizes as well as the overall standard deviation are also reported in the table. Of the studies, 16 or 84% of the study-weighted effect sizes are positive and favor the computer programming group. The range of the study-weighted effect sizes extends from -0.445 to 1.575.

When effect sizes were converted to percentiles, the overall grand mean of the study-weighted effect sizes for all 211 comparisons was 0.41. This suggests percentiles of student outcomes of 66 and 50 for the experimental and control groups, respectively. This indicates that students having computer programming experiences scored about 16 percentile points higher on various cognitive-ability tests than students not having programming experiences.

The relationship of each independent conditioning variable to the mean study-weighted effect size was tested for significance using ANOVA. Table 3 lists the F-values for the nine variables for all 19 studies. The results indicate that none of the variables had a statistically significant impact on the mean study-weighted effect size. This suggests that no one conditioning variable affected the overall student outcomes on problem solving abilities.

Discussion

The results of this meta-analysis indicate that computer programming has slightly positive effects on student problem solving performance. Eighty-four percent of positive study-weighted effect size values also confirm the effectiveness of computer-programming instruction. Yet, the findings of no significant differences for each independent variable and the study-weighted effect size suggest no single independent variable in the study has crucial impact on the positive effects of computer programming instruction. Given the small number of studies, this is not terribly surprising. Because there were only nine variables selected for the analysis of variance, further studies should explore various factors which might affect the cognitive outcomes through computer programming activities.

The results from this study, however, suggest that the outcomes of learning a computer language go beyond the content of that specific computer language. Students are able to obtain some cognitive skills such as reasoning skills, logical thinking and planning skills, and general problem solving skills through computer programming activities. Since fostering students' problem solving ability is usually part of our educational goal, the results provide to teachers a mildly effective approach for teaching students problem solving skills in the classroom setting. Left unanswered is the question of whether computer programming is as efficient, or any more efficient, at developing these problem solving abilities than other possible instructional approaches that teachers might use. Studies of this question, however, will probably require further clarification of the exact nature of the problem solving abilities most likely to be developed through programming. This meta-analysis points out only that improvements in problem solving abilities are possible, and that information by itself is useful.

References

Cheshire, F. D. (1981). The effect of learning computer programming skills on developing cognitive abilities. (Doctoral dissertation, Arizona State University, 1981). *Dissertation Abstracts International, 42*, 645A.

Clements, D. H. (1985). Research on Logo in education: Is the turtle slow but steady, or not even in the race? *Computers in the schools, 2*(2/3), 55-71.

Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. *Journal of Educational Psychology, 78*, 309-318.

Clements, D. H. (1987). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. *Journal of Educational Computing Research, 3*(1), 73-94.

Clements, D. H., & Gullo, G. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76*, 1051-1058.

Coburn, P., Kelman, P., Roberts, N., Snyder, T., Watt, D. H., & Weiner, C. (1982). *Practical guide to computers in education.* Reading, MA: Addison Wesley.

Dalton, D. W. (1986). *A comparison of the effects of Logo use and teacher-directed problem-solving instruction on the problem-solving skills, achievement, and attitudes of low, average, and high achieving junior high school learners.* Paper presented at the annual convention of the Association for Educational Communications and

Technology, Las Vegas, NV. (ERIC Document Reproduction Service No. ED 267761)

Ehrlich, K., Abbott, V., Salter, W., & Soloway, E. (1984). *Issues and problems in transfer effects of programming* (Technical Report No. 29). Paper presented at the annual meeting of the American Educational Research Association, New Orleans, LA.

Glass, G. V., McGaw, B., & Smith, M. L. (1981). *Meta-analysis in social research.* Beverly Hills, CA: Sage Publications.

Horner, C. M., & Maddux, C. D. (1985). The effect of Logo on attributions toward success. *Computers in the Schools,* 2, 45-54.

Hlawati, B. S. (1985). Effects of Logo and problem-solving CAI on the cognitive processes of gifted children. (Doctoral dissertation, Kent State University, 1985). *Dissertation Abstracts International, 47,* 402A.

Kulik, J. A., Kulik, C. C., & Bangert-Drowns, R. L. (1985). Effectiveness of computer-based education in elementary schools. *Computers in Human Behavior, 1,* 59-74.

Kurland, D. M., Pea, R., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research, 2*(4), 429-458.

Lehrer, R., & Randle, L. (1987). Program solving, metacognition and composition: The effects of interactive software for first grade children. *Journal of Educational Computing Research, 3*(4), 409-426.

Lochhead, J., & Clement, J. (Eds). (1979). *Cognitive process instruction: Research on teaching thinking skills.* Philadelphia: Franklin Institute.

Many, W. A., Lockard, J., Abrams, P. D., & Friker, W. (1988). The effect of learning to program in Logo on reasoning skills of junior high school students. *Journal of Educational Computing Research, 4*(2), 203-213.

Missiuna, C. (1987). *Development and evaluation of the "Thinking with Logo" curriculum.* Calgary Board of Education: Alberta Dept. of Education. (ERIC Document Reproduction Service No. ED 287453)

Nickerson, R. S. (1982). Computer programming as a vehicle for teaching thinking skills. *Thinking: The Journal of Philosophy for Children, 4,* 42-48.

Papert, S. (1980). *Mindstorms.* New York: Basic Books.

Pea, R. D., & Kurland, D. M. (1984a). On the cognitive effects of learning computer programming. *New Ideas in Psychology,* 2(2), 137-168.

Pea, R. D., & Kurland, D. M. (1984b). *Logo programming and the development of planning skills* (Technical Report No. 16). New York, Bank Street College of Education Center for Children and Technology. (ERIC Document Reproduction Service No. ED 249930)

Piaget, J. (1952). *The origins of intelligence in children.* New York: International University Press.

Rieber, L. (1986). *The effects of Logo on young children.* The Pennsylvania State University. (ERIC Document Reproduction Service No. ED 267788)

Rose, N. S. (1985). The effects of learning computer programming on the general problem-solving abilities of fifth grade students. (Doctoral dissertation, North Texas State University, 1983). *Dissertation Abstracts International, 44,* 2354A.

Rucinski, T. T. (1987). The effect of computer programming on the problem solving strategies of preservice teachers. (Doctoral dissertation, University of Illinois, 1986). *Dissertation Abstracts International, 47,* 3404A.

Seidman, R. H. (1981). *The effects of learning a computer programming language on the logical reasoning of school children.* Paper presented at the annual meeting of the American Educational Research Association, Los Angeles, CA. (ERIC Document Reproduction Service No. ED 205206)

Statz, J. A. (1974). The development of computer programming concepts and problem solving abilities among ten-year-olds learning LOGO. (Doctoral dissertation, Syracuse University, 1973). *Dissertation Abstracts International, 34,* 5418B.

Watt, D. (1982). Logo in the schools. *Byte,* 116-134.

Waxman, H. C., Wang, M. C., Anderson, K. A., & Walberg, H. J. (1985). Adaptive education and student outcomes: A Quantitative synthesis. *Journal of Educational Research, 78*(4), 228-236.

Table 3. F-Values for the Impact of All Conditioning Variables on the Mean Study-Weighted Effect Sizes

Variable	F-Value	df	p
Year of Publication	0.628	7	0.725
Type of Publication	1.067	2	0.367
Instructional Level	1.523	1	0.157
Language Studied	2.302	1	0.124
Number of Comparisons	0.106	1	0.749
Type of Design	0.106	3	0.995
Sample Size	0.968	2	0.401
Duration of Treatment	1.771	2	0.202
Statistics	0.514	1	0.483

Table 1. Research Descriptions

Author(s)	Source[a]	Level[b]	N	Language	Design	Duration	Tests[c]	Outcomes
Cheshire (1981)	D	S	101	BASIC	nonequivalent groups	4 mos.	DCAT	problem solving abilities
Clements (1986)	P	E	72	Logo experimental	pre/posttest	22 wks.	tdbr	cognitive development, metacognition
Clements (1987)	P	E	16	Logo	longitudinal	26 mos.	TCS	cognitive abilities
Clements et al. (1984)	P	E	18	Logo experimental	pre/posttest	12 wks.	TTCT MFFT MST	reflective, divergent thinking, cognitive development, metacognition
Dalton (1986)	E	S	54	Logo	nonequivalent	2 mos. groups	TCS TNRPS	problem solving abilities
Ehrlich et al. (1984)	E	C	82	Pascal	nonequivalent groups	4 mos.	AWP	procedural thinking skills
Hlawati (1985)	D	E	41	Logo	pre/posttest experimental	5 wks.	CFIT TTCT MFFT ME	metacognition
Horner et al. (1985)	P	E	74	Logo	nonequivalent groups	14 hrs.	GALT	problem solving skills
Kurland et al. (1986)	P	S	31	mixed experimental	pre/posttest	2 yrs.	PRT	thinking skills
Lehrer et al. (1987)	P	E	26	Logo experimental	pre/posttest	5 mos.	TOH tdbr	problem solving & metacognition
Many et al. (1988)	P	S	171	Logo	posttest-only experimental	3 mos.	NJTRS	reasoning skills
Missiuna et al. (1987)	E	E	154	Logo	nonequivalent groups	12 wks.	CCAT	problem solving abilities
Pea et al. (1984b)	E	E	32	Logo	longitudinal	1 yr.	PRET	planning skills
Pea et al. (1984b)	E	E	64	Logo	longitudinal	1 yr.	PRET	planning skills

Table 1. (cont'd)

Author(s)	Source[a]	Level[b]	N	Language	Design	Duration	Tests[c]	Outcomes
Rieber (1986)	E	E	47	Logo	nonequivalent	3 mos. groups	tdbr	procedural thinking
Rose (1985)	D	E	40	unnamed	nonequivalent	12 wks. groups	CCTT DCAT	problem solving abilities
Rucinski (1986)	D	C	99	BASIC	nonequivalent groups	4 mos.	tdbr	problem solving strategies
Seidman (1981)	E	E	41	Logo experimental	posttest-only	15 wks.	SCRT CCRT	logical reasoning
Statz (1974)	D	E	31	Logo	unspecified	1 yr.	GTQ TOH PP, WP	problem solving abilities

Note

[a] D = dissertation; E = ERIC paper; P = journal article

[b] E = elementary; S = secondary; C = college

[c] AWP = algebra word problem;
CCRT = Cornell Conditional Reasoning Test;
CCTT = Cornell Critical Thinking Test;
GALT = Group Assessment of Logical Thinking;
MFFT = Matching Familiar Figures Test;
MST = MaCarthy Screening Test;
PP = permutation puzzle;
PRT = Procedural Reasoning Test;
TCS = Test of Cognitive Skills;
TNRPS = Test of Non-Routine Problem Solving;
TTCT = Torrance Test of Creative Thinking;
CCAT = The Canadian Cognitive Ability Test;
CFIT = Culture Fair Intelligence Test;
DCAT = Developing Cognitive Ability Test;
GTQ = game of twenty questions;
ME = Markman Essays;
NJTRS = New Jersey Test of Reasoning Skills;
PRET = Plan Route Efficiency Test;
SCRT = Smith/Sturgeon Conditional Reasoning Test;
tdbr = tests developed by researchers;
TOH = Tower of Hanoi;
WP = word puzzles

Table 2. Number of Comparisons and Weighted Effect Sizes by Study

Study	Number of Comparisons	Weighted Effect Size
Cheshire (1981)	5	-0.165
Clements et al. (1984)	13	1.017
Clements (1986)	14	0.941
Clements (1987)	1	0.745
Dalton (1986)	2	0.980
Ehrlich et al. (1984)	2	0.381
Hlawati (1985)	8	0.159
Horner et al. (1985)	2	0.039
Kurland et al. (1986)	14	-0.014
Lehrer et al. (1987)	6	-0.445
Many et al. (1988)	1	0.352
Missiuna et al. (1987)	3	0.366
Pea et al. (1984b)	8	0.455
Pea et al. (1984b)[a]	108	0.273
Rieber (1986)	1	1.575
Rose (1985)	2	0.434
Rucinski (1986)	2	0.057
Seidman (1981)	11	0.128
Statz (1974)	8	0.469
Mean		0.408
Standard Deviation		0.479

Note. Total Number of Comparisons = 211

a Two separate studies are reported in this reference

LX LIVE

Sharon (Burrowes) Yoder, Chair
University of Oregon/ICCE
Eugene, Oregon

ABSTRACT

This session will present practical classroom applications of Logo. Each of four well-known columnists for the *Logo Exchange* will present several suggestions that you can use with your own students. You can hear Eadie Adamson of "Logo Ideas", Glen Bull of "Logo Connections", Judi Harris of "Logo LinX", and Gary Stager of "Stager's Stuff" present some of their favorite ideas. Applications in math, science, social studies, and language arts will be presented. Printed materials describing each activity presented during the session will be available. Don't miss this special opportunity to get to know your favorite Logo author in person!

PRESENTERS

Eadie Adamson
Allen-Stevenson School
New York, NY

Glen Bull
University of Virginia
Charlottesville, VA

Judi Harris
University of Virginia
Charlottesville, VA

Gary Stager
NAME
Wayne, NJ

SPONSOR: SIGLogo of ICCE

NEW NATIONAL DATA ON THE USE AND EFFECTIVENESS OF COMPUTER-BASED INSTRUCTION

Henry Jay Becker
Center for Research on Elementary and Middle Schools
& Center for Social Organization of Schools
The John Hopkins University
Baltimore, MD 21218

ABSTRACT

This session reports findings from two new major national studies: the first announced results from U.S. schools participating in the international survey of computer use, "Computers in Education—Worldwide, 1989"; and detailed results of a nationwide field experiment on the effectiveness of CAI in mathematics (from which preliminary findings were presented at last year's NECC).

The session is divided into two distinct presentations. In the first presentation, survey data are presented about how schools are now using computers for instruction, how their pattern of use differs from two, four, and six years ago, and what current trends and changes seem to be coming in the near future. Use patterns include types of software used, allocation of computer time across different subjects of instruction, extent of computer utilization by individual students and teachers, before- and after-school use, and arrangements for using computers during the school day.

The data come from a nationwide survey of about 1,300 computer-using schools and roughly 3,000 math, science, English, and computer education teachers, mainly from grades 5, 8, and 12. This survey was fielded between January and May of 1989, and therefore provides both timely and comprehensive information from computer-using classroom teachers and school-level computer coordinators. The survey is the third national survey in a series that goes back to 1983, but is the first to be incorporated in an international study involving nearly 20 countries in America, Europe, and Asia.

In the second presentation, newly analyzed data are presented from a 1987-88 national field experiment on computers in mathematics. In particular, the presentation will focus on the conditions resulting in greater and lesser effectiveness of CAI instructional programs in 5th through 8th grade mathematics. The data provide evidence about whether, for example, teachers' effectiveness with using computer-based math lessons and math practice varies by teacher characteristics and teaching styles, by type of student and type of school, and by hardware and software used. Effectiveness is operationally defined as differential gains in student achievement (both standardized achievement and achievement on the curriculum actually implemented) between pairs of similar classes, one taught using traditional media and one taught with a heavy emphasis of computer-based learning activity.

Because not all uses of computers are equally effective in all circumstances and with all types of students, it behooves schools to try to optimize the way they use limited computer time. These findings will help give schools evidence to support particular changes in their allocation of computer resources.

EXPLORATIONS IN SCIENCE

Eadie Adamson, Michael Tempel
Trish Koontz
Gary S. Stager

ABSTRACT: Exploring Motion through Game Programs

Eadie Adamson, The Allen-Stevenson School, New York, NY 10128; Michael Tempel, Logo Computer Systems, New York, NY.

Taking advantage of students' great fascination with controlling speed and direction of objects on a LogoWriter screen, this project was initiated to extend the possibilities for what the students might plan to create in a motion and animation context. Providing tools, simple game ideas, and giving assistance on their extensions, the project meshed the student agenda of just getting things moving with an adult agenda of providing a more "accurate" simulation of movement in a serial, two dimensional programming environment. This presentation is a brief report on what happened: the kinds of programs developed, student response to the work, observations about teaching issues which arose, and details about different ways to approach such a topic with a group of students. A paper giving a fuller description of the project and an annotated selection of procedures will be available.

ABSTRACT: Technology in Middle School Physical Science: Gender Equity

Trish Koontz, Kent State University, 404 White Hall, Kent, OH 44242.

The primary purpose of this project was four-fold: to upgrade middle school science teachers' knowledge of physics; to increase the utilization of technology during science instruction; to increase the use of manipulative devices during science instruction; and to foster an awareness of gender equity issues in science and technology instruction.

The three-phased project consisted of two physics couses, a computer course, and a practicum. The computer course emphasized the use of computers and manipulatives in science instruction. Computer interfacing probes, Lego-Logos, rocketry software, databases, and other science software available to teachers were investigated. One of the physics courses emphasized electronics and culminated in the participants building their own interface box and temperature and light probes. At the completion of the course each teacher designed a unit utilizing technology to pilot in their middle school science class.

The course outlines, examples of the teacher made units, and data on the ease of implementation of technology into the classroom will be presented.

ABSTRACT: An Inner-city Math and Science Enrichment Program Featuring LEGO ® TC logo

Gary S. Stager, Director of Training, Network for Action in Microcomputer Education, Fallon Education Center, 51 Clifford Drive, Wayne, NJ 07470.

Introduction

For twenty-one years, the Stevens Institute of Technology Science and Technology Enrichment Program (STEP) has been working to develop the math, science, and technical skills in under-represented junior high and high school minority students from target areas in the New York/New Jersey metropolitan area.

Apple Computer, Inc. donated twenty-one Apple IIgs systems to STEP for use in a math and science enrichment project featuring LEGO TC logo at the Camden Middle School in Newark, New Jersey. This project began in September of 1988.

The Setting

The seventh and eigth grade students in the project spend half of their day with a math/science teacher and half with a language arts/social studies teacher. Each math/science classoom has four or five computers and each computer station with at least one complete LEGO TC logo starter set, suitable for four students.

The Project's Goals

Improve Students' Math, Science, and Problem Solving Abilities—Our first goal is obvious. We hope to improve the students' ability to solve problems, think critically, and learn the math and science concepts necessary for life and the required standardized tests.

Restructure the Curriculum—I am working to meet the school district's rigid curriculum objectives for math and science while attempting to have the curriculum revolve around large, open-ended, child-centered, interdisciplinary theme projects. We are working towards having the language arts/social studies team teachers support these activities.

Empower the Students—LEGO TC logo provides all students with a vehicle for success—regardless of individual learning styles. We hope that our students will build self-esteem and a vision of themselves as scientists, mathematicians, inventors, engineers, computer scientists, and thereby will possess a personal link to the future.

At NECC '89, I will share examples of students' work and the curriculum units that have been developed. I will discuss the project's future, staff development, supplementary materials, curriculum integration, and the project's successes and failures. Handouts will be provided.

VIDEODISCS IN THE CLASSROOM

Edward C. Beardslee
Carole J. Meshot, Carole S. Nelson, J. Allen Watson

ABSTRACT: The Role of the Subject Matter Expert in the Development of IVD Courseware

Edward C. Beardslee, Mathematics and Computer Science, Millersville University, Millersville, PA 17551.

The task of developing a complete course of study to be delivered via IVD (Interactive Video Disc) requires the efforts and resources of many people and organizations. Though the video production, computer software development, and instructional design can be provided by industry, they typically do not have the resident expertise to define and produce the content. A subject matter expert as an outside consultant has an important role to perform in maintaining the integrity of the content as well as interfacing with various industrial disciplines.

This paper presents a model, used by the author, in developing the content for a course, the equivalent of a 3-credit college-university course. The development involved supervising twelve other writers over a nine-month time period.

This paper should be of interest to any professional educator who intends to be or has an interest in being a subject matter expert for developing IVD courseware.

ABSTRACT: "Sea Mammals": A Repurposed Interactive Videodisc-Hypermedia HyperCard Lesson for K through 2nd Grade Students

J. Allen Watson, School of Human Environmental Sciences, Department of Child Development-Family Relations, The University of North Carolina at Greensboro, Greensboro, NC 27412-5001; Carole J. Meshot, Carole S. Nelson, University of North Carolina.

For the past eight years, the Children and Technology Project (CAT) Project has been involved in research adapting computer technology for educational purposes. One of our recent, completed projects is "Sea Mammals" which is a repurposed interactive videodisc (IVD) lesson for K through 2nd grade children. The one and one-half hour lesson teaches key constructs children need to know in order to classify types of sea mammals. In our repurposing, we used a generic videodisc, an age-appropriate written script, digitized sounds, graphics, and animated scanned still frames. A secondary purpose for "Sea Mammals" was to use this sophisticated, repurposing system (IVD, HyperCard and hypermedia) to serve as a training vehicle for teachers.

"Sea Mammals" evaluation results showed: (1) that K through 2 graders are extremely attracted to hypermedia learning exercises; (2) that teachers agree that the "Sea Mammals" interactive approach could only be done by computer; (3) that children learn a larger number of conceptual constructs via hypermedia than through traditional methods (preliminary evaluation); (4) that teachers report more positive attitudes toward a clearer understanding of the direction the "change-over" might take. The public school teachers and administrators provided data to support the success of the university/public school model as a way to innovate and produce repurposed IVD and hypermedia lessons.

ART, COMPUTERS, AND THE DEVELOPING CHILD

Nancy Scali
Arroyo Elementary School
1700 East 7th Street
Ontario, CA 91764

ABSTRACT

'Children are the messages we send to tomorrow' ... technology the tools that will bridge today's children's visions in learning to a future of creative endeavors.

When children create they are making sense of the world they know and think about. Computers extend a child's creative powers by giving total control of an expressive moment and freedom in an act of communication.

Art, Computers, and the Developing Child is a special session about children's visual interpretations and communication. It will illustrate how students in the early elementary grades use computers to enhance visual thinking and extend the inventive qualities of art.

The sharing of student artwork (grades K-4) will demonstrate cooperative learning, integration of visual art throughout the curriculum, and quality art activities for the single computer classroom. Ideas for parent involvement and awareness of the value of the computer as an expressive art tool will be discussed.

A mini-gallery will exhibit examples of professional computer art, which have served as models for students to observe, study, organize, and better understand the potential of the computer as an art tool.

Provision of an on-line, off-line art activities booklet will be included in this session.

Positive moments of a child's personal visual expression support healthy self-esteem. Computers and art provide the extension for a child to develop creative powers and positive visual communication.

THE USE OF COMPREHENSIVE COMPUTER-BASED INSTRUCTION FOR THE BASIC SKILLS REMEDIATION OF EDUCATIONALLY DISADVANTAGED STUDENTS

Karen Swan, SUNY Albany
Marco Mitrani, Teachers College, Columbia University
Frank Guerrero, NYC Board of Education
John Schoener, NYC Board of Education

Abstract

The research reported in this paper investigated the efficacy of the use of comprehensive computer-based instruction for providing basic skills remediation to educationally disadvantaged student populations. Thirteen CBI programs placed in twenty-six elementary and secondary schools throughout the New York City school system were evaluated during the 1987/88 school year. Results reveal that CBI programs can indeed be an effective means for delivering such instruction, that they can be as effective in providing instruction in reading as they are in providing mathematics instruction to educationally disadvantaged students, and that within that population an inverse relationship exists between instructional level and achievement gains resulting from involvement with CBI. Interviews with participating students and teachers indicate that four features of CBI make it particularly useful to educationally disadvantaged students -- CBI is perceived by students as less threatening than traditional classroom instruction, it provides extensive drill and practice exercises, it typically provides individualized diagnostics, and CBI programs provide students with greater academic support.

Many authors have argued that comprehensive computer based instruction might best be used for delivering basic skills instruction to educationally disadvantaged students (Niemiec & Walberg, 1987). Not only is there a critical need for some means of addressing persistent functional illiteracy among an unacceptable percentage of the American population (National Commission on Excellence in Education, 1983; Commission on Reading, 1985; Kirsch & Jungeblut, 1986; National Assessment of Educational Progress, 1988), but research suggests that educationally disadvantaged students benefit most from the use of CBI systems (Jamison, Suppes & Wells, 1970; Chambers & Sprecher, 1980; Schmidt, Weinstein, Niemiec & Walberg, 1985; Niemiec & Walberg, 1987). Indications are also that students learn more quickly from CBI programs than from regular classroom instruction (Ragosta, 1982; Kulik, Kulik & Bangert-Drowns, 1985; Bangert-Drowns, Kulik & Kulik, 1985) and that their motivation for and attention to needed drill and practice is improved by the use of computer delivery (Electronic Learning Laboratory, 1982; White, 1986; Niemiec & Walberg, 1987). Such factors may be particularly important to educationally disadvantaged students. To date, however, no large scale, systematic investigations of the use of CBI for basic skills remediation among educationally disadvantaged student populations have been conducted.

Such is the work of the **Computer Pilot Program** of the Division of Computer Information Services of the New York City Board of Education. The Computer Pilot Program was designed to investigate the educational uses of computers, and to explore the efficacy of their use with New York City's educationally disadvantaged student population. The research reported in this paper is concerned with the statistical analyses of the effects of thirteen CBI programs on the reading and mathematics performance of educationally disadvantaged students in the third through tenth grades. It is unique in that it examines within a single study the instructional effects of a large sample of comprehensive CBI systems. It also looks at a very large educationally disadvantaged student population in many, very real, inner-city school settings.

Methodology

The Computer Pilot Program was funded by the Division of Computer Information Services of the New York City Board of Education in conjunction with the vendors (in the 1987/88 school year) of thirteen computer-based instructional systems -- **Autoskills, CCC, CCP, CNS, Degem, ESC, Ideal, PALS, PC Class, Plato, Prescription Learning, Wasatch,** and **Wicat.** Vendors placed their systems in appropriate schools, and offered staff development, equipment maintenance, and support at little or no cost for the evaluation period. Individual schools were responsible for staffing the programs and providing time for staff development. The Division of Computer Information Sciences provided additional equipment, technical assistance, and coordination among groups involved in the Computer Pilot Program. Evaluation of the Computer Pilot Program was carried out by the Office of Educational Assessment of the New York City Board of Education.

The goals of the Computer Pilot Program are to identify comprehensive computer-based instructional programs which can be effective in increasing the academic performance, attendance, and positive attitudes of educationally disadvantaged students, and to isolate the implementation factors significantly influencing program and/or implementation effectiveness.

During the 1987/88 school year, comprehensive CBI programs were evaluated in ten elementary, seven junior high or intermediate schools, and nine high schools throughout the New York City area. Most school sites were chosen by the Chancellor, and at each a program coordinator was selected by the school or the school district to be responsible for the daily operation of the program. Each school was also responsible for selecting a target group of students in need of remediation in basic reading and/or mathematics, and for scheduling that group in compliance with the stated needs of the particular systems. During the 1987/88 school year the citywide test scores of 1,734 students were included in the analysis of improvements in reading performance, and the scores of 1,351 students were used to analyze mathematics performance improvements.

Tests used were the **Degrees of Reading Power (DRP)** for reading achievement and the **Metropolitan Achievement Test (MAT)** for mathematics performance. Each student's 1987 and 1988 mid-instructional scores compared using several statistical analysis. Only students for whom both 1987 and 1988 scores were available were included in the study. Students in grades eleven and twelve do not take DRP's, and students in grades nine through twelve do not take MAT's, thus participating students in those grades were not included in the study. Student groupings containing less than five subjects were also omitted from the analyses in which they occurred.

Matched T-tests were used to test for significant differences between students 1987 and 1988 DRP and MAT scores, and effect sizes for the mean differences between these generated. Effect sizes were calculated by dividing mean differences by their standard deviations. Effect sizes of one or greater thus indicate performance increases of a full standard deviation or more. Matched T-tests and effect sizes were used to compare both reading and mathematics scores for computer systems (programs) by grade levels by individual schools, for grade clusters (ie. elementary, junior high, high school) by student categories (ie. special education, Chapter 1, bilingual, remedial, general).

In addition, analyses of covariance were used to compare 1988 mean scores between various student groupings (ie. grade clusters, student categories, computer systems) to test for significant differences among these. 1987 scores were used as covariates to control for individual differences.

Results

Table 1 shows the results of matched T-test comparisons broken down by systems and schools and grades for reading scores. Table 2 shows the results of matched T-test comparisons broken down by systems and schools and grades for mathematics scores. These results show that, in general, students participating in the Computer Pilot Program showed significant increases in both reading and mathematics performance. An overall effect size of 0.8 was found across in reading improvements and an overall effect size of 0.9 was found across in mathematics improvements, indicating that, in general, such increases were educationally meaningful.

Notice that these findings indicate that the CBI programs tested were, in general equally effective in producing increases in students' reading and mathematics performance. Such results argue against commonly held notions (White, 1986; Niemiec & Walberg, 1987) that CBI is a more effective means for delivering mathematics instruction than for delivering reading instruction, at least among the educationally disadvantaged student population tested.

Notice also that effect sizes decrease with advancing grade levels. Effect sizes in reading break down to 1.1 for elementary school students, 0.7 for junior high students, and 0.3 for high school students. In mathematics, effect sizes break down to 1.2 for elementary students and 0.4 for junior high students. To test for the statistical significance of this effect, analyses of covariance were used to compare the 1988 scores of students in three grade clusters -- elementary, junior high, and high school -- for reading, and two grade clusters -- elementary and junior high school -- for mathematics, using 1987 scores as a covariate to control for individual differences. Table 3 shows the results of the analysis of reading scores; Table 4 shows the the results of the analysis of mathematics scores.

The results reveal significant differences between grade clusters in both reading ($F_{1,\ 1583} = 12.93$, $p < .01$) and mathematics ($F_{1,1348} = 123.59$, $p < .01$). These findings corroborate Kulik's (1981) finding of an inverse relationship between students' instructional level and performance gains resulting from CBI use. They are particularly meaningful because they corroborate the results of his meta-analysis within a single study and with respect to an educationally disadvantaged student population.

To test the parameters of this relationship, several comparisons were made involving student categories. Analysis of covariance were used to compare 1988 mean scores in both reading and mathematics among students in five student categories -- special education, Chapter 1, bilingual, remedial, and general education -- which were understood as representing, in that order, increasingly advanced levels of instruction. 1987 mean scores were used as the covariate. The results of these analyses are given in Tables 5 and 6. They indicate significant differences in performance gain between student categories in both reading ($F_{4,1581} = 11.40$, $p < .01$) and mathematics ($F_{4,1345} = 24,74$, $p < .01$). Table 7 shows effect sizes by grade clusters and student categories for reading gains, and Table 8 shows effect sizes by grade clusters and student categories for gains in mathematics. Notice that, especially within the lower grades, an inverse relationship between instructional level and achievement can be quite clearly distinguished. Special education students benefited most from CBI use at all grade levels, and remedial and general education students benefited least at most grade levels. The inverse relationship seems clearer, however, among students in the lower grades.

To test for the significance of these seemingly different patterns of student gains, analyses of covariance were used to compare 1988 mean scores in both reading and mathematics among students by grade clusters and student categories using 1987 mean scores as the covariate (Tables 9 and 10). The results indeed reveal significant interactions between grade clusters and student categories in both reading ($F_{5,1574} = 2.92, p < .05$) and mathematics ($F_{4,1240} = 8.66, p < .01$), indicating differential patterns of achievement among student categories within differing grade clusters. Effect size data indicates that the major source of such differences involves the differential ranking of Chapter 1 and remedial students at differing grade levels. In general, however, findings show that an inverse relationship between instructional level and achievement gains resulting from CBI use holds not only between grade levels but between student categories within grade levels.

Discussion

The results of the analyses indicate that CBI can indeed be an effective means for delivering remedial instruction in reading and mathematics to educationally disadvantaged students. Significant increases in both reading and mathematics scores were found for the majority of students participating in the Computer Pilot Program. That such increases were educationally meaningful is demonstrated by their effect sizes -- 0.8 for reading and 0.9 for mathematics. These results are particularly meaningful given that they were culled from mostly start-up implementations operating within a very large and diverse educational system, and so were subject to all the constraints attendent upon such an effort. Included among such constraints were technical problems with particular implementations, a perceived lack of sufficient training among a majority of the staff participating in the program, minimal student involvement with many of the CBI programs, and, in certain cases, the newness of the programs themselves. An apparent lack of correlation between the achievement goals of the CBI programs tested and those of the City of New York, demonstrated in a regression analysis comparing students' system-embedded gains with their scores on citywide tests, may indicate another problem area. Indications are, then, that results could be even more positive as faculty and students adjust to the use of this new and different educational medium.

Our findings also suggest that CBI was as effective a medium for the delivery of instruction in reading as it was for the delivery of mathematics instruction to the educationally disadvantaged student population we surveyed. While overall effect sizes for increases in mathematics were slightly higher than those for reading performance, a larger percentage of students advanced to grade level or above in reading than in mathematics. Likewise, differences seen in interactions between content areas and other variables (ie. grade clusters, student categories, and computer systems) tended to balance themselves out. Such results contradict commonly held beliefs asserting that CBi is a more effective medium for supporting mathematics instruction than for supporting instruction in reading (Niemiec & Walberg, 1987; White, 1986). They argue instead that CBI can be an effective means for providing remedial instruction in both areas, at least among educationally disadvantaged populations.

Niemiec and Walberg (1987) suggest four reasons for CBI's particular effectiveness among educationally disadvantaged student populations. They contend that CBI is less threatening than instruction relying on classroom recitiation and that such lack of threat may be more meaningful to the educationally disadvantaged; that educationally disadvantaged students may benefit more from the extensive drill and practice exercises typically offered in CBI programs; that the diagnostic procedures integral to most such programs may benefit educationally disadvantaged students more than others because they are more likely to need specific remediation; and that extra teaching resources, which tend to be more useful to the educationally disadvantaged because they need more academic support, may be available to students involved with CBI programs. Our interviews with students and teachers participating in the Computer Pilot Program support all four arguements.

When asked how learning on computers differed from their regular classroom activities, the students we interviewed overwhelming responded that learning on computers was less threatening. "I don't have to talk," "I don't have to write," "My mistakes aren't embarrassing," " It doesn't talk back," and "The computer doesn't yell at me" were typical of the answers we received. Not only teachers, but many students we interviewed stated that the large amount of practice they received was very helpful. "The computer calls on me every time," one student commented. "You have to think more," another said. Students interviewed also believed that the immediate and informative feedback typically accompanying CBI drill and practice exercises made them especially useful. Participating teachers we interviewed believed that the diagnostic procedures offered by the CBI programs we evaluated were among their most useful features. They thought that the feedback from such procedures had made them more aware of students' individual strengths and weakness thus helping them to target specific areas where extra remediation was needed. Finally, it is unquestionably the case that extra teaching resources necessarily accompany CBI programs because the programs themselves are an important extra resource. Students reported that the programs were responsive their needs. Teachers reported that they freed them, not only from routine bookeeping and disciplinary chores, but from general lecturing as well, thus allowing them to devote more individual attention to their students. In addition, the average student/teacher ratio in the program implementations we evaluated was 15/1, considerably lower than those common in regular classrooms in the New York City system.

The results of our analyses also show significant differences between grade levels for both reading and mathematics performance gains. They thus support Kulik's (1981) assertion that an inverse relationship exists between instructional level and achievement gains resulting from CBI use. The effect sizes of student gains were greater for elementary students than for junior high students and greater for junior high students than for high school students. Such findings indicate that the inverse relationship between grade level and CBI gains which Kulik found among general student populations is likewise true of educationally disadvantaged populations.

The picture is not quite so clear when instructional level is considered in terms of student categories within grade levels. Indeed, significant interactions between grade clusters and student categories were found for both reading and mathematics. At the elementary level for both reading and mathematics, students at the lower levels of instruction (special education, Chapter1, and bilingual) experienced greater achievement score increases than students at higher levels of instruction (remedial, general.) At the junior high level for reading, special education students showed the largest performance gains and general education students showed the smallest gains, but the gains evidenced by Chapter 1 and remedial students were essentially the same. At the junior high level in mathematics, special education students again showed the greatest achievement gains and general education students, the least, but remedial students showed greater gains than those enrolled in Chapter 1 programs. The same is true of high school students' reading performance gains.

It is not clear why the patterns of relationships among student categories thus seem to change with changes in grade levels. It may be that the characteristics of students making up the Chapter 1 and remedial categories change as grade levels increase. It is interesting to note in this regard that the numbers of students entering the Computer Pilot Program at or above grade level seem to decline with increasing grade levels. On the other hand, it may be that the needs of Chapter 1 and/or remedial students differ at differing grade levels. It could be, for example, that remedial students at the upper grade levels are particularly in need of the kinds of drill and practice abundantly available in most CBI programs, whereas at lower grade levels, remedial students are in greater need of instruction concerned with other things. In any case, the findings indicate that the use of CBI was most effective for special education students and least effective for general education students at all grade levels. CBI was more effective for Chapter 1 and bilingual than for remedial students at the elementary levels, but that it was more effective for remedial than for Chapter 1 students at the upper grade levels. In general, however, they indicate an inverse relationship between instructional level and performance gains resulting from CBI use even with grade level held constant.

The explanation for this inverse relationship may simply be that CBI is a more effective delivery system for students at lower instructional levels. It is more likely that the design of current programs better supports the sort of content addressed at lower instructional levels. While the simple tutorial and drill and practice routines common to most systems seem particularly effective, not only for developing needed skills, but for diagnosing skill deficiencies at such levels, the problem solving skills inherent in higher level reading comprehension and mathematical problem solving may neither be so easily developed, nor the lack of such skills effectively diagnosed, by instructional models of this sort. In any case, the results clearly argue that the most effective use of CBI programs among the pppulation tested was in the elementary grades and among special education students, that the least effective use of them was among high school and general education students.

We can conclude, then, that CBI programs can be effectively used to deliver remedial instruction in reading and mathematics to educationally disadvantaged inner city students, that they can be as effective for delivering instruction in reading as they are for delivering mathematics instruction to such populations, and that among such populations an inverse relationship exists between instructional level and CBI effectiveness as measured in terms of achievement gains. Continued investigation of all programs at all levels is, of course, needed to better determine their relative efficacies. Other interesting questions needing further study include the long term effects of CBI use, issues of transfer from computer-based learning to other media, and changes in the structure of the overall educational environment resulting from large scale CBI use. Of particular interest are educational effects other than achievement gains resulting from involvement with CBI programs. Increased student motivation, for example, improved attendance, lower drop-out rates, and better student attitudes toward school and learning might be more beneficial to educationally disadvantaged students in the long run than short term academic achievement gains. Student perceptions of increased control over their own learly clearly deserve further study. For the present, however, the pressing realities of the crisis in education, especially among educationally disadvantaged inner city populations, together with the demonstrated efficacy of at least certain CBI programs, argue quite strongly for their adoption for the delivery of basic skills remediation to such populations, particularly at lower levels of instruction.

References

Bangert-Drowns, R. Kulik, J. & Kulik, C-L. (1985) Effectiveness of computer-based education in secondary schools. *Journal of Computer-Based Instruction, 12* (3), 59-68.

Chambers, J. A. & Sprecher, J. W. (1980) Computer-assisted instruction: current trends and critical issues. *Communications of the ACM, 23*, 332-342.

Commission on Reading. (1985) *Becoming a Nation of Readers*. National Academy of Education.

Electronic Learning Laboratory. (1982) *On Task Behavior of Students During Computer Instruction VS. Classroom Instruction*. New York: Teachers College, Columbia University.

Jamison, D., Suppes, P. & Wells, S. (1974) Effectiveness of alternative instructional media. *Review of Educational Research, 44*, 1-67.

Kirsch, I. S. & Jungeblut, A. (1986) *Literacy: Profiles of America's Young Adults* (Report No. 16-P1-02). Princeton, NJ: Educational Testing Service.

Kulik, J. (1981) Integrating findings from different levels of instruction. Paper presented at the annual meeting of the American Educational Research Association, Los Angeles.

Kulik, J., Kulik, C-L. & Bangert-Drowns, R. (1985) Effectiveness of computer-based education in elementary schools. *Computers in Human Behavior, 1*, 59-74.

National Assessment of Educational Progress. (1988) *The Mathematics Report Card: Are We Measuring Up?* Princeton, NJ: Educational Testing Service.

National Commission on Excellence in Education. (1983) *A Nation At Risk: The Imperative for Educational Reform*. Washington, D. C.: United States Department of Education.

Niemiec, R. & Walberg, H. J. (1987) Comparative effects of computer-assisted instruction: a synthesis of reviews. *Journal of Educational Computing Research, 3* (1), 19-37.

Ragosta, M. (1982) *The ETS Survey of Computer Assisted Instruction*. Princeton, NJ: Educational Testing Service.

Schmidt, M., Weinstein, T., Niemiec, R., & Walberg, H. (1985) Computer-based instruction with exceptional children: a meta-analysis of research findings. Paper presented at the annual meeting of the American Educational Research Association, Chicago.

White, M. A. (1986) Synthesis of research on electronic learning. In T. R. Cannings & S. W. Brown (Eds.) *The Information Age Classroom*. Irvine, CA: Franklin, Beedle & Associates, 17-20.

A SURVEY OF COMPUTER INTEGRATION IN THE CURRICULUM IN K-12 CENTRAL WISCONSIN SCHOOLS

Gerald E. Gau
Department of mathematics and Computing
University of Wisconsin - Stevens Point
Stevens Point, Wisconsin 54481

The results of the recently published National Assessment of Educational Progress (NAEP) suggest the need to train teachers to integrate computers in their classrooms. A survey of central Wisconsin K-12 teachers, conducted during the summer and fall of 1989, produced results that are in general agreement with the NAEP study. The survey was conducted to obtain information about how teachers use computers in their classrooms, their perception of student use of computers, and their perception of the extent to which their school district has integrated computers in the K-12 curriculum. Following is a summary of the NAEP study, a description of the Central Wisconsin survey, and a discussion of how the results will be used.

The NAEP Study

The NAEP study, which assessed the computer competence of students in grades three, seven, and eleven, was published in April 1988. It attempted to assess student competence in three areas: computer technology, applications, and programming. Bracey (Bracey, Sept. 1988) summarized the assessment and among his conclusions were the following:

> "The overwhelming majority of students in grades seven and eleven reported that they received little time in computer applications: about 50 percent never used word processing, and another 27 percent said they used word processing less than once a week. About 75 percent never used graphics and 84 percent never used a data base."

After citing other similar results he concluded with the following observation about the restricted use of computers in secondary schools:

> "Finally, the report notes that in middle schools and high schools computers are seldom used for anything except programming and other computer studies."

And quoting from the NAEP report:

> "This compartmentalization restricts students from using the computer as a general-purpose learning tool that is useful across curriculum domains."

This final quote from the NAEP study was brought resoundingly into focus with a graphic (Figure 1) which displayed student responses to the question: "Have you used a computer in any of the following classes?" With the exception of students in math classes, fewer than 25% of those surveyed had used a computer in any other class. Clearly there is room for improvement!

The Central Wisconsin Survey

While the NAEP study focused on *student* use of computers, the survey of central Wisconsin teachers focused on how *teachers* use computers with their students. The survey instrument was constructed based partly on questions taken from *A Guide to Curriculum Planning in Computer Education* (Anderson & Shires, 1987) and partly on experience gained in training teachers to use computers. It was distributed to three groups of teachers during the summer and fall of 1988, most of whom were enrolled in one of two University of Wisconsin - Stevens Point continuing education computer courses. One group consisted of 96 teachers who had enrolled in the *Word Processing, Spreadsheet and Data Base* course. These teachers typically have had little computer training. A second group consisted of 18 teachers enrolled in the *Computer Integration into the Classroom* course. This course requires previous computer experience or training as a prerequisite, so these teachers have had some background in computing. The third group consisted of 28 teachers who were participating in a summer math problem solving workshop. The computer training of this group was unknown.

The data from the entire survey were placed in three categories for analysis: elementary teachers, secondary teachers, and administrators/specialists. The major results follow:

Teacher Use of Computers

1. I assign students homework which requires the use of a computer.

	Never	Sometimes	Frequently
Elementary	88%	7%	2%
Secondary	56%	25%	19%
K - 12	71%	16%	11%

2. I use some instructional software to provide large group instruction.

	Never	Sometimes	Frequently
Elementary	59%	31%	9%
Secondary	50%	33%	17%
K - 12	54%	32%	13%

Table 1 EFFECT SIZES 1988 DRP Scores: System by School by Grade

SCHOOL		GR	N	1987	1988	X DIF	SD	ES
PRESCRIPTION LEARNING:	PS 59	4	11	34.8	40.1	5.3*	4.9	1.1
		5	20	43.3	59.6	16.3*	6.5	2.5
		6	8	49.1	54.1	5.0*	4.3	1.2
	IS 390	8	19	43.1	57.3	14.2*	4.8	3.0
CCC:	PS 152	3	18	56.0	70.0	14.0*	13.6	1.0
		4	41	31.7	44.0	12.3*	7.7	1.6
		5	44	33.3	49.8	16.5*	7.5	2.2
	PS 160	4	68	32.9	45.5	12.6*	7.9	1.6
WICAT:	PS 31	3	20	36.3	52.0	15.7*	10.8	1.5
		4	21	29.8	38.2	8.4*	6.9	1.2
		5	22	39.3	50.7	11.4*	6.5	1.8
AUTOSKILLS:	PS 246	4	71	20.6	28.7	8.1*	6.1	1.3
	IS 231	7	14	44.4	54.2	9.8*	5.1	1.9
		8	13	51.7	56.5	4.8*	6.4	0.8
ESC:	PS 332	3	21	26.2	37.4	11.2*	10.2	1.1
		4	18	21.6	28.7	7.1*	5.2	1.4
		5	17	33.1	43.9	10.8*	10.3	1.0
PC CLASS:	PS 142	3	20	29.4	45.8	16.4*	17.4	0.9
		6	24	45.0	51.7	6.7*	6.8	1.0
	JHS 141	7	10	52.4	57.2	4.8**	7.7	0.6
		8	106	57.2	63.1	4.9*	6.7	0.9
DEGEM:	PS 268	3	82	50.3	58.4	8.1*	13.3	0.6
		4	78	38.0	44.9	6.9*	6.5	1.1
		5	107	42.3	54.4	12.1*	8.8	1.4
		6	115	54.2	57.2	3.0*	6.7	0.5
	JHS 210	7	43	49.1	54.3	5.2*	7.7	0.7
		8	48	54.9	58.2	3.3*	5.9	0.6
	IS 252	7	64	49.6	54.9	5.3*	8.6	0.6
		8	26	56.6	59.4	2.8*	6.3	0.4
		9	20	56.2	58.9	2.7**	6.6	0.4
	John Jay HS	9	20	60.7	62.5	1.8	4.7	0.4
		10	77	59.9	61.9	2.0*	7.0	0.3
WASATCH:	PS 126	3	15	35.5	42.6	7.1*	9.9	0.7
		4	33	31.2	34.3	3.2*	6.6	0.5
		5	67	50.0	58.6	8.5*	8.5	1.0
		6	34	58.0	62.9	4.9*	7.2	0.7
	IS 117	7	52	50.9	52.3	1.4	11.3	0.1
		8	93	53.5	58.9	5.4*	5.9	0.9
		9	6	59.7	61.5	1.8	6.1	0.3
CNS:	JHS 189	7	37	56.8	62.6	5.8*	5.9	1.0
		9	23	57.4	60.8	3.4*	7.0	0.5
IDEAL:								
	Tilden HS	9	5	50.2	53.0	2.8	10.6	0.3
		10	17	59.2	63.2	4.0*	6.1	0.7
PLATO:	Prospect Heights HS	9	5	57.2	61.2	4.1*	7.7	0.5
		10	17	60.0	62.1	2.1	5.2	0.4
	South Bronx	9	51	57.2	59.1	1.9*	6.3	0.3
CCP:	Washington	9	22	56.0	55.5	-0.5	8.8	-0.1
	T Roosevelt	9	19	52.6	56.7	4.1*	5.5	0.9
PALS:	ML King HS	9	17	51.2	52.2	1.0	5.5	0.2
		10	6	51.7	53.5	1.8	7.1	0.3
	T Jefferson	9	10	49.8	55.1	5.3*	5.7	0.9
		10	10	58.5	49.4	0.9	6.4	0.1

* – significant difference at $p < .05$ level

** – significant difference at $p < .10$ level

Table 2 EFFECT SIZES: 1988 MAT Math Scores: System by School by Grade

SCHOOL	GR	N	1987	1988	MEAN DIF	SD	ES
CCC:							
PS 152	3	18	554.9	629.9	75.0*	19.1	3.9
	4	45	571.5	613.7	42.2*	23.6	1.8
	5	44	594.3	627.6	33.2*	19.1	1.7
PS 160	4	68	574.4	615.5	41.1*	21.1	1.9
WICAT:							
PS 31	3	20	524.9	569.2	44.3*	23.4	1.9
	4	19	554.5	590.7	36.2*	16.1	2.2
	5	23	574.6	611.0	36.3*	12.3	3.0
DEGEM							
PS 225	3	12	508.0	541.6	33.6*	23.1	1.5
	4	71	563.1	602.6	39.5*	17.5	2.3
	5	33	588.0	603.7	15.7*	17.6	0.9
	6	23	594.1	622.5	28.4*	21.8	1.3
PS 268	3	87	572.0	593.1	21.1*	33.8	0.6
	4	77	588.3	619.5	31.1*	19.7	1.6
	5	108	612.8	647.7	35.0*	23.0	1.5
	6	113	633.8	655.5	21.7*	18.0	1.2
JHS 210	7	46	617.9	628.5	10.6**	16.8	0.6
	8	43	630.8	646.8	16.0**	18.6	0.9
IS 252	7	66	617.8	621.9	4.1**	15.9	0.3
	8	21	624.2	629.1	4.9	16.5	0.3
PC CLASS:							
PS 142	3	25	512.4	568.9	56.4*	42.7	1.3
	6	28	600.9	636.7	35.8*	12.4	2.9
JHS 141	7	8	620.9	629.9	9.0	19.3	0.5
	8	91	640.1	649.0	8.9**	14.4	0.6
ESC:							
PS 332	3	22	532.8	3556.7	23.9*	26.0	0.9
	4	15	542.0	2559.7	17.7*	17.2	1.0
	5	18	587.8	4603.4	15.7*	23.6	0.7
PRESCRIPTION LEARNING:							
IS 390	8	17	602.1	621.5	19.4**	22.9	0.8
WASATCH:							
PS 126	3	17	554.4	560.0	5.6	21.0	0.3
	4	35	564.1	579.4	15.3*	22.1	0.7
	5	65	617.2	642.5	25.2*	21.8	1.2
	6	34	642.7	670.6	28.0*	19.7	1.4
IS 117	7	55	626.5	621.4	-5.0	122.4	-0.2
	8	88	621.4	633.6	12.2**	14.5	0.8
CNS:							
JHS 189	7	39	656.6	653.4	-3.2	20.8	-0.2

* -- significant difference at p < .01 level

** -- significant difference at p < .05 level

Table 3 ANCOVA TABLE*
1988 Citywide Degrees of Reading Power (DRP) Scores by Grade Clusters

	SS	DF	MS	F	P
W/IN CELL	84244.76	1583	53.22		
REGRESSION	148676.19	1	148676.19	2793.70	.000
GRADCLUST	1375.96	2	687.98	12.93	.000

*1987 DRP reading scores as covariate

Table 4 ANCOVA TABLE*
1988 Scaled Metropolitan Achievement Test (MAT) Math Scores by Grade Clusters

	SS	DF	MS	F	P
W/IN CELL	587073.01	1348	435.51		
REGRESSION	1481310.94	1	1481310.94	3401.29	.000
GRADCLUST	53823.15	1	53823.15	123.59	.000

*1987 MAT math scores as covariate

Table 5 ANCOVA TABLE*
1988 Citywide Degrees of Reading Power (DRP) Scores by Student Categories

	SS	DF	MS	F	P
W/IN CELL	83221.31	1581	52.64		
REGRESSION	176310.06	1	176310.06	3349.46	.000
STUDCAT	2399.42	4	599.85	11.40	.000

*1987 DRP reading scores as covariate

Table 6 ANCOVA TABLE*
1988 Scaled Metropolitan Achievement Test (MAT) Math Scores by Student Categories

	SS	DF	MS	F	P
W/IN CELL	596979.00	1345	443.85		
REGRESSION	1180858.49	1	1180858.49	2660.49	.000
STUDCAT	43917.16	4	10979.29	24.74	.000

*1987 MAT math scores as covariate

Table 7 EFFECT SIZES
1988 DRP Scores by Grade Clusters and Student Categories

STUD CAT	N	1987	1988	MEAN DIF	SD	ES
ELEMENTARY:						
Special Ed	10	32.8	46.9	14.1*	6.6	2.1
Chapter 1	289	31.6	40.2	8.6*	7.4	1.2
Bilingual	46	37.6	49.0	11.4*	9.6	1.1
General	256	50.9	58.3	7.4*	8.6	0.9
Remedial	104	38.9	45.9	8.0*	8.7	0.8
JUNIOR HIGH:						
Special Ed	19	43.1	57.3	14.2**	4.8	3.0
Chapter 1	266	51.6	56.5	4.9*	7.4	0.7
Remedial	241	54.9	59.8	4.9*	6.9	0.7
General	44	59.6	62.3	2.7	9.7	0.3
HIGH SCHOOL:						
Remedial	21	57.9	61.3	3.4*	4.9	0.7
Chapter 1	296	56.4	58.6	2.2	6.8	0.3

Table 8 EFFECT SIZES
1988 MAT Math Scores by Grade Clusters and Student Categories

STUD CAT	N	1987	1988	MEAN DIF	SD	ES
ELEMENTARY:						
Special Ed	11	567.5	604.1	36.5*	25.6	1.4
Chapter 1	371	565.8	595.8	30.0*	22.5	1.3
Bilingual	58	578.9	610.4	31.4*	26.3	1.2
General	296	621.1	649.5	28.4*	25.3	1.1
Remedial	128	582.9	606.1	23.2*	23.1	1.0
JUNIOR HIGH:						
Special Ed	17	602.1	621.5	19.4**	22.9	0.8
Remedial	202	632.4	642.0	9.6*	16.7	0.6
Chapter 1	242	619.9	626.7	6.7*	17.1	0.4
General	41	653.4	648.2	-5.2	25.2	-0.2

* -- significant difference at p < .01 level
**-- significant difference at p < .05 leve

Table 9 ANCOVA TABLE*
1988 DRP Scores by Grade Cluster and Student Catgory

	SS	DF	MS	F	P
W/IN CELL	82162.10	1574	52.20		
REGRESSION	102165.11	1	1021165.11	1957.20	.000
GRDCLUST	190.46	2	95.23	1.82	.162
STUDCAT	189.18	4	47.30	.91	.460
GC X SC	761.41	5	152.28	2.92	.013

*1987 DRP reading scores as covariate

Table 10 ANCOVA TABLE*
1988 Scaled MAT Math Scores by Grade Cluster and Student Catgory

	SS	DF	MS	F	P
W/IN CELL	560357.78	1340	131.63		
REGRESSION	975156.66	1	975156.66	2331.92	.000
GRDCLUST	1350.77	1	1350.77	3.23	.073
STUDCAT	1878.54	4	469.64	1.12	.344
GC X SC	14525.65	4	3631.41	8.68	.000

*1987 MAT math scores as covariate

3. I use some instructional software to provide small group instruction.

	Never	Sometimes	Frequently
Elementary	22%	47%	29%
Secondary	36%	45%	17%
K - 12	30%	46%	23%

4. I use a word processor.

	Never	Sometimes	Frequently
Elementary	35%	31%	33%
Secondary	31%	31%	38%
K - 12	33%	31%	35%
Admin./Spec.	40%	20%	35%

5. I keep records/grades on a computer.

	Never	Sometimes	Frequently
Elementary	74%	16%	9%
Secondary	52%	9%	38%
K - 12	62%	12%	24%
Admin./Spec.	60%	20%	10%

Student Use of Computers

1. My students use drill and practice software.

	Never	Sometimes	Frequently
Elementary	9%	35%	53%
Secondary	39%	47%	14%
K - 12	25%	41%	33%

2. My students use science and/or social studies simulation software.

	Never	Sometimes	Frequently
Elementary	59%	31%	5%
Secondary	63%	31%	5%
K - 12	61%	31%	5%

3. My students use a word processor for writing assignments.

	Never	Sometimes	Frequently
Elementary	50%	35%	12%
Secondary	33%	55%	11%
K - 12	41%	45%	12%

4. My students use data base software to manipulate information in curriculum data bases.

	Never	Sometimes	Frequently
Elementary	85%	7%	2%
Secondary	81%	13%	3%
K - 12	83%	10%	3%

5. My students use graphing software to help them understand math functions.

	Never	Sometimes	Frequently
Elementary	83%	14%	0%
Secondary	70%	25%	2%
K - 12	76%	20%	1%

School District Implementation of Computers

School District Computer Integration Plan

Response of K-12 teachers on existence of implementation plan:	None:	12%
	1/3:	29%
	2/3:	11%
	Done:	14%
	Don't know:	30%

The survey results suggest that teachers are making efforts to use the computer in their instruction. Nearly half of the teachers surveyed use instructional software at least some of the time for large group instruction and about two-thirds of them use it for small group instruction. Some of them also are using it to write tests, lesson plans, (66% use a word processor) and keep records (38%). However, the fact that 88% of the elementary teachers surveyed and 56% of the secondary teachers surveyed *never* assign their students homework involving a computer indicates that most students are not using the computer as a tool to help them with the learning process. The exception may be in the area of writing where 57% of the students use a word processor at least some of the time. These results appear to agree with those of the NAEP which found that fewer than 25% of students surveyed use a computer in most of their classes. However, they appear to disagree with the NAEP with respect to which students are using the computer. The NAEP results suggest more use of the computer by elementary students. The central Wisconsin results suggest more use by secondary students.

Two other areas indicating some computer use by students were drill and practice and simulations. The fact that more than 74% of the students (88% elementary and 61% secondary) use drill and practice software is not surprising given the widespread availability of this kind of software and the ease with which it can be used. This high use of drill and practice software might account for the NAEP finding of more use of computers by elementary students than secondary students. The results indicating student use of simulations (36%) are a little more surprising given that fewer than 15% of the students in the NAEP survey used a computer in science or social studies classes. Student manipulation of curriculum data bases (13%) and the use of graphing software (21%) seem to agree with the results reported by the NAEP.

As for school district leadership: 12% of the teachers surveyed stated that their districts did not have a computer implementation plan and 30% didn't know of plans for implementation. It would seem that many school districts are not providing leadership in this area.

Intended Use of Survey Results

The literature clearly indicates there is a *need* for computer integration in the curriculum. Bracey's summary of the NAEP study indicates that there is very little student use of computers in all curricular areas with the exception of mathematics. And even in this area fewer than 40% of the secondary students surveyed used the computer. Dossey et.al. (Dossey, Mullis, Ina, Lindquist, & Chambers, 1988) reported that most computer use is by students with higher mathematical ability. The central Wisconsin survey data support the need also, given the fact that 70% of the teachers surveyed never assigned homework that required the use of a computer.

What should be done?

The National Council of Teachers of Mathematics (NCTM, 1988) recommends that course content be modified and that teachers learn how to help students use computers as tools for problem solving, exploration of new concepts, drill and practice, and in the transition from the concrete to the abstract. Roblyer (Roblyer, 1988) reviewed research which suggested that simulations are particularly effective and that word processors improve student attitudes toward writing. Other educators have suggested specific lists of computer integration objectives. McCarthy (McCarthy, 1988), for example, lists ten. Among the ten were the following: computers should be used in areas where traditional techniques have not been successful; areas where computers have worked should be identified; ask teachers where they need help and then see if the computer can help; the role of the teacher may have to change; teachers must be convinced that computers will help students learn.

The central Wisconsin survey addressed one of McCarthy's objectives: Ask teachers how they use the computer and how they would like to use it. Many of the teachers in the survey stated the following needs: Revise a unit of instruction to include an instructional software package. Become comfortable using the computer with students.

Based on the above discussion and the practical experience gained by helping teachers learn to use computers over the past eight years, the following objectives and activities seem appropriate. They are the major goals of our *Integrating the Computer into the Classroom* course and will be the focus of a summer workshop on computer integration in the curriculum.

Computer Integration Objectives

1. Review and/or develop skill in the use of the major productivity tools: wordprocessing, spreadsheet, database, graphics, and desktop publishing.
   a. Review/teach WP, SS, DB, graphics, and desktop publishing software. Group participants according to interest and need.
   b. Each participant will complete a project in two of the productivity software areas.
2. Teachers will describe how computers are being used in their classroom. They will identify effective and ineffective computer use and areas in which they would like to use the computer to improve their instruction.
   a. Each teacher will keep a logbook of use or intended use of the computer in the classroom. Each will identify units which could more properly be taught with a computer.
   b. The needs identified in the logbook will be addressed in the course.
3. Teachers will evaluate their district's plan for integrating computers in the K-12 curriculum.
4. Software objectives:
   a. Locate and evaluate simulation, tutorial, drill and practice, instructional game, lab analysis, instructional management, and other instructional software.
   b. Determine where student written computer programs can be used to enhance learning. The learning should be the focus, not the program.
   c. Locate and evaluate templates and databases that can be used with productivity software.
5. Develop a unit of instruction that will help students as they explore new concepts, move from the concrete to the abstract, problem solve, or practice skills.
   a. Locate appropriate software. Have each participant send for preview software appropriate to their needs. Examine the software provided by state agencies. Examine software available in their school districts. Examine software available in the UWSP library. Examine the Wisconsin DPI computer curriculum guide for evaluation suggestions.
   b. Develop lead-in activities that establish a need for a given software package to help solve some problem.
   c. Develop activities that teach students to use the software package.
   d. Develop activities that have students use the software package to solve some problem.
   e. Try out the unit with students. Video tape it.
6. Practice using the computer as a demonstration tool to assist students as they explore new concepts, to help them move from the concrete to the abstract, to assist them with problem solving, and to help them practice skills.
   a. Each teacher will teach a lesson from his computer integration unit to the course participants.
   b. The participants will critique the lesson.
   c. The lesson will be video taped so that the teacher participant can do a self-evaluation.
   d. Each teacher participant will demonstrate one software package.
7. Learn to use curriculum data bases. Learn to use on-line or CD-ROM research tools such as an on-line library card catalog, ERIC, and the Wilson Databases.
8. Find ways to use the wordprocessor in all curricular areas.
   a. Examine professional journals for writing activities in all curricular areas.

   b. Examine and use writing analysis tools.
   c. Build writing activities into the computer integration unit.
9. Examine different hardware environments for a classroom. A lab. A school. A district.
   a. Describe and use a microcomputer network.
   b. Discuss and demonstrate ways to effectively use a single computer in a lecture hall, with a standard class, and with a small group.
   c. Discuss the pros and cons of department computer labs versus school computer labs.
   d. Visit some schools and examine their hardware facilities.

References

Anderson, Elaine M. and Shires, Leslyn M. (1987). *A Guide to Curriculum Planning in Computer Education* (pp. 51-7). Madison, Wisconsin: Wisconsin Department of Public Instruction.

Bracey, Gerald W. (Sept. 1988). Computing at Risk: Students Get Bad Grade on First Report Card. *Electronic Learning*, (pp. 36-7).

Bracey, Gerald W. (Oct. 1988). Computers and Learning: The Research Jury Is Still Out. *Electronic Learning*.

Dossey, John A., Mullis, Ina V. S., Lindquist, Mary M. and Chambers, Donald L. (1988). *The Mathematics Report Card: Are We Measuring Up? Trends and Achievement Based on the 1986 National Assessment* (p. 91). Princeton, New Jersey: The Educational Testing Service.

McCarthy, Robert (Sept. 1988). Making the Future Work: The Road to Curriculum Integration. *Electronic Learning*, (p. 43).

National Council of Teachers of Mathematics (1988). *HANDBOOK, 1988-89: Association Leaders and Position Statements* (p. 16). Reston, Virginia.

Roblyer, M. D. (Sept. 1988). The Effectiveness of Microcomputers in Education: A Review of the Research from 1980-1987. *T.H.E JOURNAL*, (p. 88).

Figure 1: Computer Use in Subject Areas

```
                           Percent
           0--5--10--15--20--25--30--35--40--45--50--55
Math     3 *******************************************
         7 ******************************
        11 **********************

Reading 3 ********************
English 7 *******************
        11 *************

Science 3 *********
         7 ********
        11 **********

Social   3 ********
Studies 7 *******
        11 ****

 Art     3 *****************
         7 *****
        11 **

Music    3 *************
         7 *****
        11 ***
```

THE FUTURE OF EDUCATIONAL COMPUTING

Lawrence Lipsitz
Editor, *Educational Technology*
Educational Technology Publications
720 Palisade Avenue
Englewood Cliffs, NJ 07632

ABSTRACT

Following ten full years of intense activity in the field of educational computing, we are at a crossroads. This session will explore where we have been in the application of computer technology in education and where we might go in the future. This future is not set in microchips but is under our control; thus the decisions we make in the next few years become of paramount importance.

The microcomputer revolution of the1980s has been a most curious long-term happening in that despite enormous gains the impact of computing on the daily life of schools and students has been and remains minimal. Schools today differ little from the institutions of a decade ago, and many would claim that in fact they are worse today.

Can the role of computers in the schools be redirected in order to serve as change agents rather than supporters of traditional educational practices and procedures? What might be the dimensions of such a redirection of effort? What are the implications for research? For teacher training? For software development? For overall educational design? Can the educational computing movement become allied with the ongoing school reform movement? What kind of educational computing do we really want, and how can it be integrated with other emerging educational technologies? These and related matters will be examined in this speech, to be followed by audience participation and discussion, aimed at bringing about a broader conceptualization of educational computing and educational technology in general.

Homework for Attendees: Spend some time thinking about your personal *vision* of educational computing ten years from the date of this conference, 1999. What is your preferred model of educational computing? What is your preferred model of schooling? Do these models coincide? And how to get there from here? Come to the session prepared to discuss such matters. Don't worry: There will be no grades given.

CRITICAL NON-TECHNICAL SKILLS FOR COMPUTER COORDINATORS AND ORGANIZING INSTRUCTIONAL TECHNOLOGY EFFORTS IN LARGER SCHOOL DISTRICTS

Marilyn Gardner, Doreen Kelly
Boston Public Schools
75 New Dudley Street
Boston, MA 02119

ABSTRACT

This session focuses on the extraordinary skills necessary to successfully manage the job of a Computer Coordinator in public education in the 1990s. It identifies educable skills one needs and can attain to be an effective Computer Coordinator. The presentation will take the practical and earned experiences of a Computer Coordinator and a Technology Support Specialist in a large urban school district. They will mold their experiences into a presentation of valuable insights and advice for programmatic success in organizing instructional technology expansion efforts in larger school districts.

The general outline is as follows:

I. The Everyday Skills
II. What Relationship With The Business Community?
III. What Do Great Organizations Have in Common?
IV. Characteristics of a Good Leader
V. The Effective Computer Coordinator
VI. When Should I Assert Leardership?
VII. Leadership Myths
VIII. The Unpleasant Side of Leadership
IX. How to Organize and Develop System Wide Initiatives
X. At What Price Excellence?

LIBRARY AUTOMATION AT UNION COLLEGE: THE BROADER PERSPECTIVE

David V. Cossey, Chair
Union College
Schenectady, NY

ABSTRACT

Selection of a library automation system usually takes place after an extensive analysis of library needs in such areas as cataloging, circulation, acquisitions, serials control, and public access. The Schaffer Library Automation Committee at Union College (Schenectady, NY) broadened its view and explicitly considered as well the campus computing environment. This environment consisted of a multitechnology communications network, a VAXcluster for academic computing and two Hewlett-Packard systems for administrative computing.

The automation committee decided that integration of the library system into the campus network would provide a number of advantages, including ease of access for users, the potential of interfacing with other campus offices, and backup in case of failure of the library computer.

Balancing these benefits would be the need for relinquishing some autonomy and for recognizing the possible complications involved in working closely with the computer services staff to coordinate planning, implementation, and functioning of the chosen system. Viewing the library as part of the broader campus environment may result in a different system choice than if library needs are the sole concern, but the resulting system will better support the overall mission of the library and the college.

SPEAKERS

Jean K. Sheviak
Union College
Schenectady, NY

David V. Cossey
Union College
Schenectady, NY

SPONSOR: ASCUE

TEACHING ETHICS IN COMPUTER SCIENCE

Lillian N. Cassel, Chair
Villanova University
Villanova, PA

ABSTRACT

The importance of ethics questions at all levels of education has become increasingly apparent. Computer science is concerned with such questions as privacy, rights to data and to ideas expressed in programs, vulnerability of critical systems, responsibility and accountability for correctness of procedures and accuracy of data. This panel will address the question of raising ethics issues with students at high school and college levels.

PANELISTS

George G. Bear
University of Delaware
Newark, DE

John Beidler
University of Scranton
Scranton, PA

Paul Gormley
Villanova University
Villanova, PA

Harriet Taylor
Louisiana State University
Baton Rouge, LA

RESPONDER

Richard Austing
University of Maryland
College Park, MD

SPONSOR: SIGCSE

EXPERT ADVISING SYSTEMS IN A COLLEGE/UNIVERSITY ENVIRONMENT

Gerard Kiernan
Arnold Koltun
Edward N. Schwartz

Manhattanville College
Department of Mathematics and Computer Science
Purchase, New York 10804

Abstract

In the last five years expert systems have been constructed in a variety of areas.[1] It is interesting to note that although expert systems are most appropriate for problems involving diagnosis and advising there are not many examples of expert systems written for diagnostic and advising problems in the academic area. In this paper, we describe two expert systems that we have constructed that give academic advice. In a more general context, we consider the problems and the possibilities for developing expert systems in an academic environment.

Introduction

Several years ago we decided to learn about expert systems. We read as much as we could find about this area of artificial intelligence but we discovered that we didn't really have a grasp of the theory, let alone the practice, of design and implementation of an expert system. We decided that we could learn a great deal by actually creating an expert system. We hoped to integrate the information that we got from various sources with some concrete, practical knowledge.

We realized that the actual programming of a system from scratch in a language like Prolog or LISP would be a truly daunting task. We decided that a sensible approach would be to do our work using an expert system shell that would free us from the tedium of generating questions, screen management, et cetera and would allow us to concentrate on the problems of knowledge engineering and of programming the system.

We soon realized that the selection of a problem was of crucial importance. We needed a problem that we could handle. We needed an available expert. We needed a problem that was amenable to solution with an expert system. After searching for such a problem, we realized that there were many such problems "in our own backyard".

Some of the most successful expert systems are those that do diagnosis and/or give advice. We realized that in an academic environment we were often in the position of giving advice to students and, that the advice often was formulated on the basis of some sort of expert diagnosis. In an academic setting there are often many scattered pockets of expertise. Unfortunately, there is usually no mechanism by which these pockets can communicate with one another. One of the rationales for writing an expert system is to capture expertise so as to be able to distribute it. In searching the literature, we found that businesses use expert systems because they provide a useful and profitable tool. We found that the same was not true in an academic setting. We felt that this was an unfortunate lack and that since we knew about the academic area, it was a likely place for us to begin.

Problem selection

The question of how to recognize a problem suitable to solution by an expert system is somewhat difficult. We have found that since an expert system tries to capture the expertise of a domain expert in some relatively restricted area, a problem suitable for solution by expert system should, first of all, be solvable by an expert. That sounds silly but there are many problems that are so big, or so fuzzy that there are no recognized experts and, therefore, it would be very hard to discern the heuristics that are used by an expert in their solutions. Another characteristic of a problem that is ripe for solution by an expert system is that such a problem should have reasonably well-defined solutions. That is to say, it should be possible to recognize when a problem has been solved, otherwise it will be very hard to validate any expert systems written to solve the problem.

Problems should also be reasonably limited in scope. What this means is very hard to quantify. We have found that expert systems are, in some ways constructible in a modular way. That means that a very large problem can often be attacked by breaking it down into sub-problems. These sub-problems, then, must be manageable. Another characteristic of problems that are amenable to expert system solution is that they generally have a finite set of well-defined possible solutions. Generally, the components of these solutions should be enumerable in a reasonable way. If a problem has, essentially infinitely many possible components of any solution, it is not likely to be a good candidate for an expert system.

Historically, diagnostic problems have been considered ripe for expert system solution. Among the first successful expert systems were the various medical diagnostic systems like MYCIN and EMYCIN. These systems help to diagnosis an illness, given answers to certain questions. There are many more such systems in use today. Advising problems are in some sense

---

1.see [6]

61

isomorphic to diagnostic problems. In an advising problem, one is given some set of conditions (such as a student's class, major, and courses taken). From these conditions some sort of advice, say on courses to take, is generated by the expert. This is essentially the same procedure as doing a medical diagnosis. Problems such as these are much more amenable to rule based, backward chaining type methods than are "synthesis" problems. Here we start with a finite, known set of possible recommendations. In a synthesis problem, entirely new conclusions might be drawn by combining information.

Initially, we were looking for a problem that would allow us to study the techniques involved in the construction of an expert system. As we examined possible problems, we realized that the final product could be valuable. Our expert systems both solve advising problems. We have written a system that gives academic advice on course selection to mathematics and computer science majors at Manhattanville College. This past summer we wrote a system that gives advice to incoming students on the appropriate choice of an initial mathematics course for them. While developing these expert systems, we needed to devise methods for representing the rules and documenting the system. These technical details are described elsewhere.[2.] In this paper, we focus on what the systems do and on more general development questions.

Choice of software

When creating an expert system one must, rather early on, decide upon the method by which the information will be programmed into the computer. There are many possible approaches. One might decide to invent an entirely new language or tool to solve the problem. This is done [3.]. It is very interesting but certainly very difficult and not something that someone interested only in a final working end product would want to do. Another possibility is to program in some existing upper level language. Once again, this presents some very interesting and very hard problems. A third possibility is to use an expert system shell. An expert system shell is a tool that allows the knowledge engineer to program an expert system at the level of the rules. The tool generally provides an inference engine, screen management, question generation, and explanation facilities.

We decided that we would use an expert system shell to build our system. We wanted a system that would run on an IBM-AT. We wanted a shell that would allow us to compile the expert system for improved performance and so that a stand alone system could be created. We also wanted a shell that would allow us to write sections of the system in some upper level programming language to allow for certain custom features. When we started our work two years ago, there were not a great many expert system shells that met our criteria. We chose the Arity Expert System Development Package and the Arity Prolog Compiler to do our work. On the whole, we have been quite satisfied with our choice.

Finding and working with experts is hard!

We have found that finding and working with experts is a very hard task. In our experience experts are not much interested in describing what they do in detail and are even less willing to try to abstract the heuristics that they employ. In fact, we found that often an expert will be adamant about something that she says she does, only for us to discover later that this is not really what she does at all.

Our first system, the academic advisor for mathematics and computer science majors, was highly anomalous because we served as both the knowledge engineers and also as the experts. During the creation of this system we found that we could not, simply, interview ourselves. We found that we had disagreements about the heuristics for giving advice and that we even had disagreements about the advice that should be given in certain circumstances. We were able, however, to clarify our advice and our own heuristics during the sometimes rather heated discussions that we had about the advice that should be given under various circumstances. This is certainly a desirable outcome but, in general, is not something that one would expect during an interview with an outside domain expert.

Our second system was only a bit more standard in its form. This system had a true outside expert. For the past five years we have had someone at the college whose task it is to assess each incoming student and to make a determination as to the appropriate mathematics course for this student to take. Unfortunately, since we are all members of the Mathematics Department, we all thought that we had a better way to give appropriate advice than our domain expert. We made a conscious decision to "implement the expert" as a first approximation to our advice giving system. After we had a good implementation of our expert, we added our own heuristics. We tested to see whether our initial implementation was good by running the information available to the expert on last year's entering class through the system. We were satisfied when most of the advice that was given by the system matched the advice given by the expert. We found that our first implementation of the expert was good but not as good as we had hoped. This happened, we discovered, because very often the expert was adamant about some heuristic or other and yet, when we looked at the actual advice that was given it turned out that this heuristic was not the one that was used. Once we realized this, we decided to implement the expert subject to the advice that she actually gave, rather than only subject to the rules that we were able to elicit from her in a series of interviews.

We believe that the process described above is one that must be examined carefully. The process of constructing an expert system will often force the experts to understand more clearly and explicitly how they reach their conclusions. With our first system we

---

2.see [2],[3],[4],[5]

3.see [1]

decided that we were the appropriate experts and that we could and should implement the heuristics that we felt to be suitable. In the course of the development of this system, we found that our advising heuristics were both clarified and made more uniform. With our second system we decided that we would simply listen to the expert. Nevertheless, we eventually created a richer system by adding our own heuristics. It is our suspicion that the knowledge engineer will always be tempted to impose heuristics that seem appropriate, even when designing a system in a domain about which he or she knows nothing. We suspect that this could be a disastrous practice.

Our next enhancement of the advising system for incoming students will be to give advice on the appropriate course in English Writing. In this version of the system we will truly be dealing with an area about which we are supposed to know nothing. We hope that we can resist the temptation to impose our own rules on the expertise of the English Department.

Knowledge engineering

Knowledge engineering is the most difficult part of the creation of an expert system. The process of discovering the heuristics employed by the experts is difficult. This is true precisely because the expert uses rules of thumb, rather than hard and fast formulae for a final problem determination. Often an expert is not sure what these rules of thumb are. Frequently in an interview an expert will say something like "...you just know." While this may be true, it is very hard to program. Moreover, an expert will frequently assure the knowledge engineer that something is a hard and fast rule. Some time later, however, the expert will explain a counter-example to the rule. The knowledge engineer must be very careful at times like these not to alienate the expert by pointing out the inconsistency. That inconsistency is often instructive upon later reflection but is maddening when it appears.

Another problem that frequently arises is that the expert often believes that certain heuristics are used when, in fact, entirely different rules are really applied. Experts also often seem to use information that they are not aware that they are using. When pointed out to them, we have had the experience that half the time the expert will say such information is not used and the other half the time the expert will say that obviously that information must be used. Yet another set of problems arise when there are multiple domain experts. All too often, each of these experts uses very different rules to reach the same goal. It becomes extremely difficult to sort this type of knowledge into usable categories.

In dealing with the expert it must be understood by both the knowledge engineer and the expert that the creation of an expert system is a continuing process. The initial interviews are really just the beginning. Usually the expert and the knowledge engineer need some time to digest the discussion before the next interviews, at which time lots of changes are likely to be made. Once the knowledge has been programmed, the expert must help to test the system and to refine it. This is often a long process.

A purely practical problem arises from the necessity of writing down the outcome of interviews and writing down the heuristics in a compact form. This problem is further compounded by the need to program the system at some technical level. One wishes to have an overview of the system. Yet, when confronted by a mass of written rules it is very difficult to see what is going on. We have attacked this problem by developing a tree-like notation[4.] that allows us to write the rules in a compact form. Our notation also allows us to order the rules in a logical way. In fact, we have found that this notation allows us to program the expert system at a higher level than even the level of the shell.[5.] We now think of programming the system at the level of these trees, rather than at any lower level.

While there are reported to be expert systems in business that do valuable work with only about fifty rules, our experience has been that even in a very strongly constrained problem like ours, lots more rules are required to make the fine distinctions that we wish to make. Additionally, we feel that it is important to take care of anomalous cases, which uses up many rules. Each of our systems, which we have worked hard to optimize, contains about three hundred rules. As we refine the systems, the number of rules tends to grow. We often start construction with a zeroth order system that has few rules. This system is then refined and, possibly discarded, until a satisfactory working system has been produced.

Summary of the expert system construction process

Software selection

Some basic questions will guide the selection of software used to write the expert system. On what computers will the final system run? Will the system be an interactive system requiring fast response or will it run in batch mode? At what level are you willing or able to program? An expert system can be written in any language but some languages, like LISP and Prolog for instance, seem to be more appropriate for this task. An expert system shell will hide some of the programming details, ease screen handling and the generation of explanations. The best advice is to read reviews and talk to people who are writing expert systems.

Problem selection and working with experts

A suitable problem should be limited in scope and have a finite (reasonably small), well defined set of possible solutions. The expertise involved in solving this problem should be valuable to non-experts. The

---

[4.] see [3]

[5.] see [5]

experts must be agreeable to, and available for, some lengthy interviews and reviews of successive systems as these systems are generated.

Knowledge engineering

The process of discovering the heuristics is slow and complicated. Use some method to record rules that is flexible, easy to use, and easy to change. Start with a small system and refine it.

Brief description of our systems

Manhattanville College is a small, private, coeducational, liberal arts college located in Purchase, New York. The programs of the college include twenty three academic majors. In addition to a major area of study, in which a student must complete a specified set of requirements and a senior thesis, each student is expected to complete a second and a third area of study. Moreover, each student is expected to submit a portfolio to the Board on Academic Standards, prior to graduation, which demonstrates proficiency in areas such as written critical analysis, quantitative analysis, and bibliographic research.

Every first year student is enrolled in a section of a large course, called a preceptorial, in addition to more traditional small courses. The student's preceptor (teacher of that student's section of the preceptorial) serves, among other things, as the student's academic advisor. A student does not usually declare a major until the end of his or her sophomore year. At that time the student may change to an advisor from the major department, may retain his or her original advisor, or may choose any other faculty member as an advisor. In addition, at the end of the sophomore year, each student must submit his or her portfolio for first review by the Board on Academic Standards. The portfolio submitted must include, among other paperwork, a complete study plan and rationale for the study plan. A study plan is, essentially, a list of all of the courses that the student will take, and the semester in which they will be taken, during the student's stay at the College.

A problem that is immediately apparent with this system is that a student's advisor may have very little knowledge about that student's proposed major field of study. Although most students eventually switch to advisors in their major, advice in the first several years is often not as good as it might be. A further problem is that academic advising is usually not a straight-forward task. While the information regarding prerequisites that appears in the college catalog is usually correct, it is not always complete. Moreover, information about prerequisites is often necessary, but is rarely sufficient to determine what courses should be taken and the order in which the courses should be taken. Furthermore, it is the policy of the college to keep listed prerequisites as sparse as possible. As advisors, we all have developed our own heuristics and biases regarding what to take and when to take it.

The first system that we built was designed to give academic advice to Mathematics majors and to Computer Science majors. The system asks the student for information about his or her major and his or her year. It also asks about courses that the student has taken. It then generates a set of recommendations for the student as to the courses in mathematics and computer science for which the student should register. This system has several uses. It allows upper level students to get advice in preparing their programs before they see one of us in the Department. It allows Sophomores to get help in constructing their study plans. They simply have the system project the courses that would be needed in subsequent semesters. It also allows prospective majors to see what is actually involved in completing one of the two majors offered by our department. The system works quite well. In some cases, the accuracy of its advice is eerie. It has been compiled and runs quite satisfactorily on a 4.77mHz PC. The only problem is that the compiled, running system requires over 600 K of disk space on a single disk. It is not, therefore, usable on a PC with only 360K floppies.

Our second system addresses the problem of the appropriate initial mathematics course for an incoming student. The system asks a variety of questions about the student's academic interests and performance in high school. Based on the answers, it generates a recommendation from a variety of possible courses and availability of tutoring options. The system is still in the testing stage but it seems to give very accurate recommendations. This system has also been compiled. The compiled, running system requires 700 K of disk space on a single disk. In addition it does a great many disk accesses. As a result its operation is fairly slow. We intend to turn this system into a batch, rather than an interactive, system. That is, we intend to write Prolog goals that will allow the system to query a large file containing the appropriate information about the incoming students. The system will then generate a file of recommendations. These recommendations will be printed and distributed to the students and to their advisors. We also intend to purchase a 25mHz 80386 system that will allow both systems to run at a significant increase in speed.

We are quite satisfied with the operation of both of these systems. We believe that there is place in an academic environment for expert systems but the role that the play should be carefully studied. We have never considered, nor does it seem wise to do so, the possibility that all academic advising should be computerized. The model of each student using an expert system as the only source of academic advice is not our goal. We see the expert system as part of a computer assisted advising system where each advisor can have access to the local areas of expertise that are scattered throughout the college or university.

References

[1] Golumbic, M.C., Markovich, M., Tiomkin, M., "A Knowledge Representation Language for University Requirements", IBM Israel Scientific Center Haifa, Israel

[2] Kiernan, G., Koltun, A., Psihountas, G., Schwartz, E., "Manhattanville College Expert Academic Advisor-A First Look", *Proceedings of the National Educational Computing Conference*, 1987

[3] Kiernan, G., Koltun, A., Schwartz, E. "Constructing an expert system - Software Engineering of a Different Kind" pp. 223-228 *CSC'88 Proceedings*, Association for Computing Machinery, 1988

[4] Kiernan, G., Koltun, A., Psihountas, G., Schwartz, E., "Some Techniques for Minimizing and Optimizing the Rule Base of an Expert System", *CSC '88 Proceedings*, Association for Computing Machinery, 1988

[5] "Programming Expert Systems at the K-Tree Level", Kiernan, G., Koltun, A., Schwartz, E. *CSC'89 Proceedings*, Association for Computing Machinery, 1989

[6] Waterman, D., *A Guide to Expert Systems*, Addison-Wesley, Reading, Mass. 1986 pp. 40-48

WHAT HAPPENS IF YOU INTRODUCE AN INTELLIGENT TUTORING SYSTEM IN THE CLASSROOM: A CASE STUDY OF THE GEOMETRY TUTOR

Yasmin Kafai
Department of Electrical Engineering and Computer Science
Advanced Technologies Laboratory
University of Michigan

In the last decade several Intelligent Tutoring Systems have been developed. The goal of Intelligent Tutoring Systems (ITS) is to help people acquire expertise through individualized learning. These systems have reached a level of functioning which allows evaluation of their effectiveness. This report describes the introduction of an intelligent tutoring system into a classroom setting. We used the Geometry Tutor developed by Anderson et al. (1985) at Carnegie Mellon University in three classes to help the students do congruence proofs. We considered two major aspects in our evaluation: the transportability of the ITS into the classroom and the tutor's instructional effectiveness. The results of this study indicate that problems may arise in integrating an ITS into the curriculum. These problems have implications on the instructional effectiveness of intelligent tutoring systems. However, most students volunteered when working with the Geometry Tutor to do as many proofs as possible. Taking advantage of this aspect may indicate possible positive and empowering uses of the Geometry Tutor.

Introduction

Individualized learning has been shown to be more effective than methods of group instruction (Bloom 1984). This suggests that even suboptimal individualized pedagogical strategies may prove a more powerful educational advance when implemented via Intelligent Computer Assisted Instruction (ICAI). Various domains have been covered: Arithmetic skills (WEST Burton & Brown 1982, DEBUGGY Burton 1982), Geometry (Geometry Tutor Anderson, Boyle & Yost 1985), Programming (BRIDGE Bonar & Weil 1985, PROUST Johnson & Soloway 1986, LISP-Tutor Anderson & Reiser 1985), Electronics (QUEST White & Frederiksen 1985, SOPHIE Brown, Burton & DeKleer 1982), and Engineering (STEAMER Williams, Hollan & Stevens 1981). The approach taken is to create intelligent tutors or coaches to replicate many of the advantages of a private human tutor. Intelligent computer based tutors have the ability to work interactively, to follow what a student is trying to do, to diagnose difficulties the student is having, and to provide instruction relevant to that difficulty.

So far, most projects have concentrated on theoretical and technical issues of ICAI and have dealt more with the internal evaluation of these systems (Littman & Soloway 1987). Internal evaluation focusses on the relationship between the architecture of an ITS and its behavior. It tries to determine whether the ITS performs in the way the designers intended it to do or what knowledge the ITS incorporates. Little attention has been paid to the other important aspect of the evaluation process: the external evaluation. External evaluation seeks to determine the impact of the ITS on the students learning. Some of these systems have been exposed for a short time to 'real students' or in classroom settings (Table 1) with positive effects such as time savings in learning and better test results. But, only a few studies undertook such an external evaluation of their ITS (Baker et al. 1985, Sack et al. 1988).

One of our concerns in this paper is not only to undertake the external evaluation of one such an ITS, but also to broaden explicitly the definition of external evaluation. Besides the instructional effectiveness we additionally consider aspects such as the transportability and adaptability of an ITS in the classroom setting. Implicitly these issues have been part of the discussion, but treated under the seperate cover of pedagogical issues. We will demonstrate with our approach the need for an integrated evaluation.

Motivation and Goals

The system we evaluated was the Geometry Tutor (GT) designed by Anderson et al. (1985) at Carnegie Mellon University. Our choice was guided by practical motivations: there is a partial version of the GT available for the Macintosh and we had a classroom set up with nine Macintoshs SE's available. In this study it is not our concern whether the GT functions adequately on a theoretical level or if the GT is better than traditional methods of teaching. Our focus is to investigate whether effective learning with the tutor is possible and whether the GT is transportable to the classroom. We address the following issues:

Instructional effectiveness: One of the critical tests of an ITS is "Can it teach students what it is suppose to teach them?" An ITS stands or falls upon the ability to perform well in face of the pragmatic challenges of the classroom. We investigated the performance of the GT in instructing students and examined the students ability to use it effectively. Related to this aspect is that the GT uses a different proof method from textbooks to do proofs, i.e. the proof path (PP) method. The traditional method used in textbooks is called the statement-reason (SR) method. Were the students able to learn this new proof method when interacting with the GT? What were the specific problems of students using the proof path method? Additionally we integrated the students in the evaluation process. So we were able to analyze the students performance with the GT, and also to ask them to evaluate specific features of the GT.

Transportability of the system into the classroom setting: The issue of transportability includes such aspects as the teachers' lack of familiarity with the material and the machinery, logistical and practical impediments to computer usage, and the lack of guidelines for the integration of the systems. These aspects constitute actual problems in the classroom reality as Schofield and Verban (1987) indicated. Since the current technology is leaving its familiar surroundings (Dede 1986), unforeseen problems may arise. To explain this situation we might think of the different 'lives' an ITS has. The first life of the ITS deals essentially with the design of the system and making it run. This phase takes place in the laboratory. The second phase of the ITS concentrates on the instructional effectiveness. Here the system moves under the control of the research team into the classroom and students/users work with the system. The third and so far neglected life of the ITS takes place the moment when the system leaves the familiar surrounding and is available for other schools which are out of immediate reach of the research team. Here the teacher takes the role previously played by the research team and has to handle the machines, provide introduction and fit the machines into his/her curriculum.

The questions we tried to answer: how can these systems be introduced into the traditional classroom setting, and what are the problems which one may encounter from teachers and students? What kind of introduction method need to be supplied? How does the system fit conceptionally into the curriculum of geometry classes? What are the instructional differences (textbook vs tutor)? We provided an introduction method, the workshop, which seemed to be appropriate for the present version of the GT.

We start with a description of the Geometry Tutor. The next step is to describe the study we undertook at a public high school with three geometry classes - two normal and one advanced. We then present an overview of how we introduced the Geometry Tutor and the results of that introduction. Finally we examine and discuss the benefits of the GT.

Description of the Geometry Tutor

The Geometry Tutor (GT) was designed by Anderson, Boyle, & Reiser (1985) to assist high school students in constructing proofs in Euclidian plane geometry. A number of principles directed the construction of the Geometry Tutor. We will discuss two of these principles because of their educational relevance here: the use of the proof path, and the instruction in context/immediacy of feedback.

Use of the Proof Path: According to the authors, "*a major effort in the design of the interface has been to communicate to the student the logical structure of a proof and the structure of the problem solving process by which a proof is generated* " (Anderson et al. 1985, p.4). The result is a proof path presented on the screen (Fig.1,2,3,4) which includes the rules/definitions applied to the premises during the inferences in generating the proof. The student grows the path by a combination of pointing to statements on the screen and typing in information. The advantages of this method are that it illustrates the structure of a complete proof and the critical features of the proof.

Instruction in Context and Immediacy of Feedback: Students appear to learn information better if they are confronted with information in the problem solving context (see Fig.1) rather than in instruction apart from the problem solving context. The GT provides immediate feedback on the students problem solving efforts. Whenever the students make a logically incorrect step of inference the system responds by identifying the logical error.

Different help facilities are available for the student:
- *Student seeking information*

Windows can be opened to see the set of rules and definitions covered so far and a more detailed explanation of these can be displayed on the screen.
- *Student seeking help*

In the command menu the **Explain** function allows the student to ask for additional help to continue the proof. Even if the chosen premises are not the most highly rated rule in the state of the proof by the tutor, the system will choose an adequate set which allows the student to continue. The student is offered at every trial a graduated hint (Fig.1). A graduated hint means to give the student help first in a very general way and to become more specific with every additional step.
- *Student getting help/information*

After three unsuccessful trials the tutor will do the step for the student.

A PC version of the GT is available on a Macintosh SE. It offers the student a set of 17 different congruence proof problems of varying difficulty. The GT provides help to the student through constant monitoring and structuring of the students problem solving attempts in the process of generating a proof.

In the following figures we give an example of how a student does a congruence proof in the Geometry Tutor. The student chooses one of the problems and receives on the screen the initial problem state (Fig.2).

The statement to be proven is displayed at the top of the screen, the givens are displayed at the bottom and a diagram in the upper left-hand corner. In the present version the student can only reason forward from the givens, applying rules and conclusions. Proof problems depending on their degree of complexity can require several proof steps to trace a set of inferences to the top (Fig. 3 represents an intermediate problem state). The final proof shows the complete path for the solution (Fig. 4) .

Introduction of the Geometry Tutor in the Classroom

The School and the Students

Three geometry classes (in total 27 students) of a public high school participated in this study. Two classes had average geometry students and one was a honors class in geometry. All classes had covered nearly the same material in geometry, essentially plane euclidian proofs of congruency, although the honors class was much farther through the curriculum. The geometry teacher, the same for of all classes, agreed to allow his classes to participate for one week at the Geometry Tutor Workshop during their normal class meeting.

The introduction to the Geometry Tutor did not take place in the classroom but in a workshop. One of the computer rooms in the school were set up with 9 Macintosh SE's. All of the computers were connected via a network system to a printer.

Description of the Geometry Tutor Workshop

The workshop was designed to make students familiar with the Macintosh and Geometry Tutor features, to enhance what students learn from normal geometry instruction and to teach new geometry rules not covered in the average geometry classes. The workshop was taught instead of the normal instruction and took place on five consecutive one period lessons in one week. The Geometry Tutor Workshop was divided into three different phases:

- the *Introduction Phase* when the students were taught how to use the Macintosh and the GT.
- the *Learning Phase* when every student worked at his/her own pace on a number of different proof problems and did homework either in the familiar statement-reason or in the proof-path method.
- The *Testing Phase* included a test for students, a questionnaire about computer attitudes, and an evaluation of the GT.

Some of the instruction and testing materials are presented in more detail along with the results. Further information can be found in Kafai (1988).

Results

The results are presented as follows: first we give an overview of the effects and impressions of the GT workshop and students. Then we evaluate the GT performance using two criteria: the instructional effectiveness and the transportability of the GT into the classroom.

Overall Performance

Summarizing the impressions, we can say that the use of the GT was a success in many regards for all three classes. In total 27 students participated in this study and all of them recommended the GT for other students. All students completed the minimal problem set and most of them did all of the additional 7 problems[1]. The students showed a surprisingly positive attitude towards computers regardless of their sex, class or geometry background. The average score on the computer attitude questionnaire for all students was 89 points (s=6.9) on a scale of 100 which indicates a fairly positive attitude. Apparently, computers are more familiar for students today. This is especially true in this school, since most students take computer related classes. We might further conclude that whatever the effects of the Geometry Tutor are, they will at least not be influenced by a negative attitude towards computers. Furthermore, the difference between the normal and the advanced geometry classes proved not to be significant. For this reason we decided to aggregate both data sets and all forthcoming analyses will refer to this one data set.

Instructional Effectiveness

Because the evaluation of the instructional effectiveness of the GT is particularly difficult, we approach this task from different sides. On one hand we evaluate the results of the actual performance of the student using the GT, on the other hand we use the students performances when applying the proof path method to do congruence proofs.

Homework

During the GT workshop we gavé the students three assignments of two problems each to do at home. For every problem we told them what proof method to apply (see figure 5). One major characteristic of these homeworks was that all of the proofs had been done before by the student in class on the GT, so that the students were already familiar with the proof problem and structure. At the end of every proof we asked the students if they had problem doing the proof and which method they thought were easier for them to do the proof.

Looking at the percentages of correct solutions, we can see that the performance of the students decreases remarkably when using the proof path method. This low performance is related to the particular character of the tasks which will be described in more detail in section "Transportability of the system". If the proof structure in the GT is not similar to the that of the statement-reason method, the students have more difficulties (86% and 62% in Homework 2 and 3). If the proof path is similar to the way the proof is done in the familiar statement-reason format, the difficulties decrease (29% in Homework 2). That the percentage is still relatively high might indicate the lack of familiarity with the proof method. No matter what method of proof the students were been assigned, 2/3 of them thought that the actual proof was easier to do in the statement-reason method.

Reconstruction of the Proof Process

A direct comparison between both proof methods: statement-reason, and proof path, can be drawn from the reconstruction task. Every student had 4 minutes to complete a simple geometry proof. The students in two classes (one normal geometry class and one advanced geometry class) were randomly assigned to two groups. One had to reconstruct a statement-reason solution, the other had to reconstruct the right order of the proof path

---

[1] Remark from the Geometry teacher: *"It never happened to me that the students were asking for additional problems ."*

(see Figure 6). In order to rearrange the proof path correctly the student had to draw arrows from the givens to the rule to apply and then to the conclusion.

As the results indicate the performance in the SR groups is significantly higher than that in the proof path method (Chi-Square = 20.16, df= 1, $p < 0.01$). We can conclude from that result that the proof task itself was not a problem, but the assigned method which refers to the results from prior analyses such as the homework.

A closer look at the students erroneous proof path solutions reveals that the two major difficulties are first that the students reverse the order of statements and reasons in the given proof problem. Most of the students who produced only a partially correct proof path solution, drew the arrows in the way they would arrange the statements and reasons in the SR-method. A students solution supporting this observation is shown in Figure 7 below.

The circled areas indicate erroneous reconstructions. For example, in area I the arrows are not drawn from the premises (=statements) to the rules (=reason), but to the conclusion first. In fact, looking closely at the generation of a proof path, a third part has been included in the proof - the conclusion of applying rules to premises. So every proof step in the proof path method consists of three parts. This is different from the statement-reason method where two parts are written out and the third is used to be the premise for the next step.

The second major difficulty is that the students do not keep the sequence of steps in the right order, although the sequence of proof steps is the same as in the statement-reason method. Nevertheless, it is probably a result of the lack of familiarity with the proof path method. Circled area II and III give an example of how proof steps can be set in the wrong order. Instead of drawing the arrows from all three premises to the rule SSS, an arrow was drawn from the premises to the conclusion of applying the rule SSS.

The evaluation answers after the task indicate that the students think that the SR method is still the preferred one, though 2/3 of the students doing the proof path think the proof is understandable in both ways. This answer is somehow in contradiction with the performance of the students. It might address a different issue that although the students still do not master this new method, they consider this method to be worthwhile approach.

Transportability of the System

The usual evaluation of ITS concentrates on two aspects: the internal and external evaluation. The external evaluation focuses on the different aspects of educational effectiveness. We think that an additional aspect, the transportability of the ITS, is not taken into account in this traditional evaluation approach. The next section describes how the Geometry Tutor and the current curriculum matched each other.

The matching of the GT and the curriculum

One of the authors' claim was *'because it does not do lesson presentation and because new problems are easy to generate, the tutor can be easily adjusted to fit any curriculum'* (Boyle 1987, p.1). If this claim is true then the students should not have any problems working with the GT. In fact, the way proofs were taught in the textbook differed in form and content from the way the proofs were done in the GT. In one problem the GT (Figure 9) required three additional steps before arriving at the same conclusion as the way presented in the textbook (Figure 8).

Most students (86%) thought that this was a fairly difficult problem for them due to the additional steps required. The students were in a similar situation with another proof problem. Comparing the results of the GT session and the homework, there was only a slight decrease in errors (from 86% to 80%) and the number of incorrect solutions speaks for itself.

Discussion

What are the results of this study? One aspect concerned the instructional effectiveness; we tried to answer the question whether the GT could teach the students geometry proof skills in the proof path method. We are not in a position to answer this question definitely since our results are ambiguous and we still did not cover the whole field to be tested. If the student had to apply outside of the GT the new proof method such as in the reconstruction situation, the performance was poor compared to the performance with the familiar SR method. A closer look at the errors revealed that most of them were due to the students confusion between the methods. Additionally, the GT required the students to be more explicit in their proof sequence. The percentage of perceived difficulties in this situation is very high. A comparison between the problems perceived at the actual interaction with the GT (86%) and the Homework 2 of this same task (80%), shows that there is no decrease. Apparently, it would take the student much more time to learn to integrate these additional proof steps. This could be seen as an indicator that the transition between the methods is not as smooth for students as one would think.

As for the transportability of the system, we can positively say that nearly all of the students enjoyed using the Geometry Tutor and would recommend it for other students. When the students were confronted with a difficult problem, the majority of them chose to do it with the GT. But we should not forget that at the end of the workshop about half of the students did not think that the GT was better to use to do proofs. Contrast these observations with a difficult integration into the classroom. There was a mismatch in at least two places (one reported in section 4.3.1) between the way the proof was done in the Geometry tutor and the method explained in the textbook. With a greater number of problems in the GT (there were only 17) one might imagine a greater number of these kind of mismatches. On the whole, the 'paper-pencil' test of the Geometry Tutor was not very successful. We understand the 'paper-pencil' test as a means to investigate what happens when the machinery is not available to the students.

At this point let us have a closer look at some of the GT features: the quality of help commands and the user interface. The GT got a good rating (AM=3.6, s=0.7) on a five-point scale for giving helpful hints on the proofs. We asked about the quality of the hints: 82% thought that the hints were easy to understand, 96% that they were appropriate for the problem and 87% that they provided enough help. Another question concerned to what extent the Tutor helped them to feel more comfortable with doing proof, the students answered that they felt quite comfortable (AM=3.5, s=1.2). It will be difficult to explain the students' contradictory answers. Although half of the students thought that using the GT was not better in doing proofs, the majority of them qualify the help provided by the GT as good. One explanation might be that the unfamiliar proof method overweights the students general impressions. Another aspect, the temporary 'rigidity' of the GT, appeared quite often during the actual interaction with the GT. Consider the following situation: a student decides to apply in the final phase of the proof the AAS rule to his/her givens. After the Inference dialog box appears on the screen, the student changes his/her mind and wants to apply the rule SSS (which in the described proof situation was possible since the student had all the necessary givens). But the student will be unable to leave the inference because the GT *thinks* that the student is on the right path. This is true but it neglects the possibility that in a given situation the user might decide that another proof step might be more appropriate. The described situation puts the user in a situation where s/he experiences helpnessless. Nothing can be done to go one proof step back. Several students experienced this situation at different steps in the proof.

In follow-up study (Kafai, in prep.) the Geometry Tutor will be used to explore alternative proof solutions. The students will be invited to reason about disadvantages and advantages of chosen solutions. We will investigate whether this helps the students to acquire better proof path skills, but also if it improves their reasoning skills. Additionally we will have a closer look at the tutorial intervention of the GT. Therefore we will record the actual interactions of the student with the GT and analyze the effect of the tutor's help. Results are forthcoming.

References

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science, 228*, 456-467.

Anderson, J. R., Boyle, C. F., & Yost, G. (1985) The Geometry Tutor. *Proceedings of the International Joint Conference on Artificial Intelligence*, Los Angelos, pp.1-7.

Baker, E,. Feifer, R. Aschbacher, P., Bradley, C., & Herman, J. (1985). *Intelligent computer-assisted instruction (ICAI) study*. Los Angelos, CA: Center for the Study of Evaluation, University of California.

Bloom, B. S. (1984) The 2 Sigma problem: the search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, vol. 13, pp. 4-16.

Boyle, C. F. (1987) *Geometry Proof Tutor User's Manual*. Carnegie-Mellon University, Pittsburgh, PA.

Bonar, J. G., & Weil, W. (1985) An informal programming language. Paper presented at the meeting *Expert Systems in Government*, October, Washington, D.C.

Brown, J. S., Burton, R. R., & de Kleer, J. (1982) Pedagogical natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman, & J. S. Brown (Ed.), *Intelligent Tutoring Systems* (pp. 227 - 282), London: Academic Press.

Burton, R. R. (1982) Diagnosing bugs in a simple procedural skill. In D. Sleeman, & J. S. Brown (Ed.), *Intelligent Tutoring Systems*, London: Academic Press.

Burton, R. R., & Brown, J. S. (1982) An investigation of computer coaching for informal learning activities. In D. D.Sleeman, & J. S. Brown (Ed.), *Intelligent Tutoring Systems* (pp. 79-98) London: Academic Press.

Dede, C. (1986) *A review and synthesis of recent research in ICAI*. International Journal of Man-Machine Studies, vol. 24, pp. 329 - 353.

Johnson, W. L., & Soloway, E. M. (1984) PROUST: knowledge-based program debugging. *Proceedings of the Seventh International Software Engineering Conference*, Orlando, Florida, pp. 369-380.

Kafai, Y. B.(1988) *The introduction of intelligent tutoring systems into the classroom setting: a case study of the Geometry Tutor*. Research Report #61, Cognition and Programming Project, Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor.

Lewis, M. W., Milson, R., & Anderson, J. R. (1987). The TEACHER'S APPRENTICE: Designing an Intelligent Authoring System for High School Mathematics, in Kearsley, G. P. (Ed.), *Artificial Intelligence & Instruction*, Reading, MA: Addison-Wesley Pub. Co., pp. 269 - 301.

Littman, D. & Soloway, E. (1987) Evaluating intelligent tutoring systems: The cognitive science perspective, in J. Jeff Richardson, M. Polson (Eds.), *Intelligent Tutoring Systems*, Hillsdale, NJ: Erlsbaum.

Rhoad, R., Milauskas, G., & Whipple, R. (1984) *Geometry* (Revised Edition). Evanston, Il: McDougal, Letell & Company. pp. 81

Schofield, J. W. & Verban, D. (1987) *Computer Usage in the Teaching of Mathematics: Issues which need answers*. Unpublished paper. University of Pittsburgh, PA.

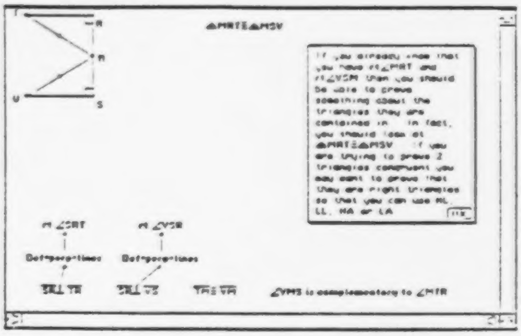
Wenger, E. (1987) *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Los Altos, CA: Kaufmann Pub. Inc..

White, B. Y., & Frederiksen, J. R. (1985) QUEST: qualitative understanding of electrical system troubleshooting. *ACM SIGART Newsletter*, no.93, pp.34-37.

Williams, M. D., Hollan, J. D., & Stevens, A. L. (1981) An overview of STEAMER: an advanced computer-assisted instruction system for propulsion engineering. *Behavior Research Methods and Instrumentation*, vol.13 (2), pp. 85-90.

systems	exposure
EXCHECK & LOGIC course	extensively used in university courses
BIP	used in college-level introductory programming courses
SOPHIE-I	available over an electronic network (ARPA) for two years
SOPHIE-I & II	used in short electronic courses
WEST	used in an elementary school for 4 months
BUGGY	game played by teachers in training; used also by students
DEBUGGY	has diagnosed problems for thousands of students
STEAMER	used in classroom at Great Lake Naval Training Center
PROUST	used in some college introductory programming courses
GUIDON-WATCH	used in a medical school class
LISP tutor	used in college-level introductory programming courses; available commercially
GEOMETRY tutor	used for a term in a high-school class

Table 1 Some experimentations with systems in actual use (Etienne Wegner, 1987)


T
R
M
U
S
△MRT≅△MSV
rt ∠SRT
rt ∠VSR
Def-perp-lines
Def-perp-lines
SR⊥TR
SR⊥VS
TR≅VM
∠VMS is complementary to ∠MTR

If you already know that you have rt∠MRT and rt∠VSM then you should be able to prove something about the triangles they are contained in. In fact, you should look at △MRT≅△MSV. If you are trying to prove 2 triangles congruent you may want to prove that they are right triangles so that you can use HL, LL, HA or LA

Figure 1 Getting Help from the Tutor

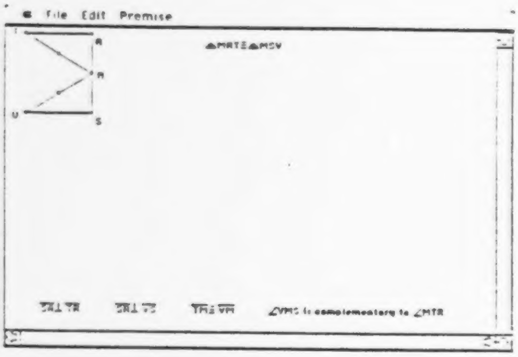

File
Edit
Premise
T
R
M
U
S
△MRT≅△MSV
SR⊥TR
SR⊥VS
TR≅VM
∠VMS is complementary to ∠MTR

Figure 2 Initial Problem State

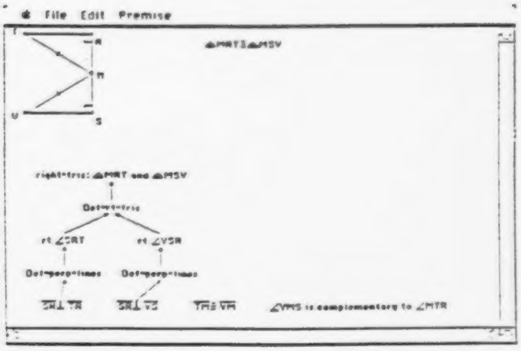

File
Edit
Premise
T
R
M
U
S
△MRT≅△MSV
right-tris: △MRT and △MSV
Def-rt-tris
rt ∠SRT
rt ∠VSR
Def-perp-lines
Def-perp-lines
SR⊥TR
SR⊥VS
TR≅VM
∠VMS is complementary to ∠MTR

Figure 3 Intermediate Problem State

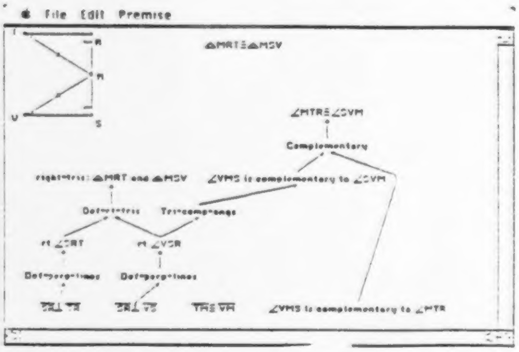
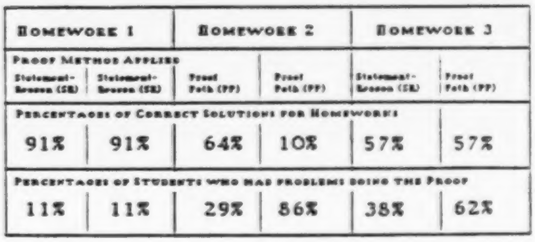
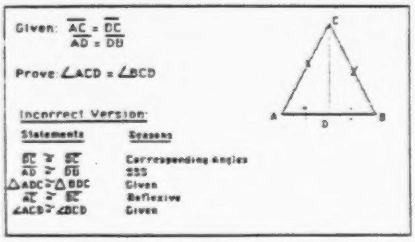

File
Edit
Premise
T
R
M
U
S
△MRT≅△MSV
∠MTR≅∠SVM
Complementary
right-tris: △MRT and △MSV
∠VMS is complementary to ∠SVM
Def-rt-tris
Tri-comp-angs
rt ∠SRT
rt ∠VSR
Def-perp-lines
Def-perp-lines
SR⊥TR
SR⊥VS
TR≅VM
∠VMS is complementary to ∠MTR

Figure 4 Final Problem State



HOMEWORK 1		HOMEWORK 2		HOMEWORK 3	
PROOF METHOD APPLIED					
Statement-Reason (SR)	Statement-Reason (SR)	Proof Path (PP)	Proof Path (PP)	Statement-Reason (SR)	Proof Path (PP)
PERCENTAGES OF CORRECT SOLUTIONS FOR HOMEWORKS					
91%	91%	64%	10%	57%	57%
PERCENTAGES OF STUDENTS WHO HAD PROBLEMS DOING THE PROOF					
11%	11%	29%	86%	38%	62%

Figure 5 Results of homework assignments



Given: $\overline{AC} = \overline{BC}$
$\overline{AD} = \overline{DB}$

Prove: $\angle ACD = \angle BCD$

Incorrect Version:

Statements	Reasons
$\overline{DC} \cong \overline{DC}$	Corresponding Angles
$\overline{AD} \cong \overline{DB}$	SSS
$\triangle ADC \cong \triangle BDC$	Given
$\overline{AC} \cong \overline{BC}$	Reflexive
$\angle ACD \cong \angle BCD$	Given

Figure 6a Problem assignment for statement-reason (SR) task

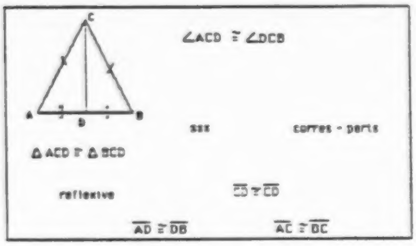

C
A
D
B
∠ACD ≅ ∠DCB
sss
corres - parts
△ACD ≅ △BCD
reflexive
CD ≅ CD
AD ≅ DB
AC ≅ BC

Figure 6b Problem assignment for proof path (PP) task

Statement-Reason (SR)	Proof Path (PP)
87%	37%

Table 2 Percentages of correct solutions

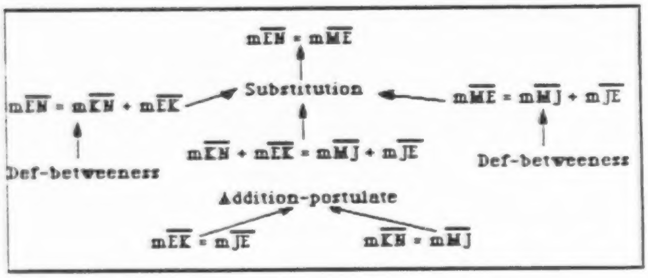

mEN = mME
mEN = mKN + mEK
Substitution
mME = mMJ + mJE
Def-betweeness
mKN + mEK = mMJ + mJE
Def-betweeness
Addition-postulate
mEK = mJE
mKN = mMJ

Figure 8 Proof Steps in the Geometry Tutor

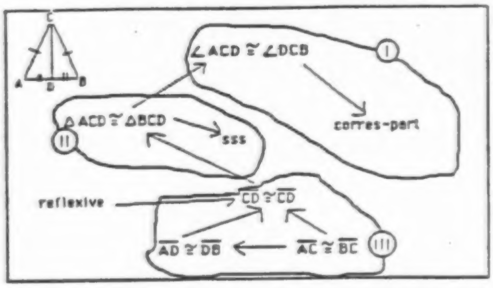

C
A
D
B
∠ACD ≅ ∠DCB
I
△ACD ≅ △BCD
II
sss
corres-part
reflexive
CD ≅ CD
AD ≅ DB
AC ≅ BC
III

Figure 7 A student's solution to the proof path task

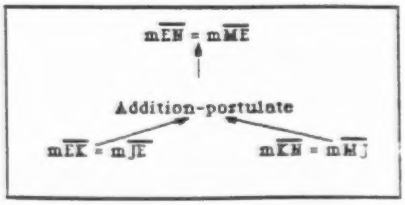

mEN = mME
Addition-postulate
mEK = mJE
mKN = mMJ

Figure 9 Proof Steps in the textbook

How difficult was this problem for you?	
- fairly difficult	86%
- less difficult	14%
How did you overcome the difficulties?	
- tutor help	69%
- thinking hard	28%
- did not overcome	8%

Table 3 Students' evaluation of problem difficulty

THREE DIFFERENT APPROACHES TO USING PROBLEMS WITH THE GEOMETRIC SUPPOSER

Daniel Chazan
Education Development Center
55 Chapel St.
Newton, Ma 02160
(617)-969-7100 x315

Richard Houde
Mathematics Department
Weston, Ma

ABSTRACT: This paper presents three approaches for using exploratory (lab) problems with the GEOMETRIC SUPPOSER. It is hoped that this paper will stimulate debate about how the SUPPOSER can be used in classrooms.

The SUPPOSER is a tool for geometrical inquiry, it is not an instructional package. Like any open-ended instrument, it has no built-in tasks or prescribed methods of use. It has a set of capabilities whose utility and power are only evident in the context of a geometrical task or a problem. Therefore, there are as many approaches to using the software as there are ways of posing problems.

As a first cut, one can imagine a spectrum of approaches that use the GEOMETRIC SUPPOSER in activity settings which differ in the quantity of exploration that students do. Some teachers might use the GEOMETRIC SUPPOSER primarily as an electronic blackboard. Others might lead their class as one group through the inquiry process, at every point soliciting suggestions from members of the class. Yet, others might allow students to explore problems individually or in small groups.

Electronic Blackboard	Whole Group Exploration	Small Group Exploration

Fig. 1 These three approaches lie on a continuum. Movement from left to right represents "freer," less constrained exploratory activity.

We distinguish among a range of uses in the most exploratory part of this spectrum, where students are expected to explore a problem on their own. We will describe three types of exploratory pedagogies. The distinctions between these three types of pedagogy are sometimes hard to delineate. After all, these three types of pedagogy are closely related. However, a differentiating factor between the three types is the role that the conjectures that the teacher expects play in the teacher's development of the course. The different roles that these conjectures play are reflected in the structure of the given problem. Each approach to exploration has characteristic problem types.

We will examine these pedagogical approaches in order from the most constrained exploration to the freest one, from "Guided Exploration," through "Application," to "Beyond the Teacher's Knowledge." We will provide examples of each approach and indicate the characteristic types of problems that go with each approach. The examples used in our discussion are standard classroom problems, ones that could be used in a high school geometry course.

Small Group Exploration

Guided Exploration	Application	Beyond the Teacher's Knowledge

Fig. 2 These three approaches lie on a continuum. Movement from left to right represents "freer," less constrained exploratory activity.

Guided Exploration

Some of the expected conjectures are vital to the teacher's agenda.

In this approach, the teacher has a stake in the students' exploration because they hope that the students will discover a relationship that is of central use later on in the course. A rationale for this type of guided exploration is that students will remember ideas that they discover on their own better than ideas that are told to them.

We will distinguish between two types of guided exploration problems. These two types differ in the type of assistance that the teacher provides to the student to aid them in discovering the important conjecture. Note that in any such problem students may come up with other conjectures that are not directly related to the teacher's agenda.

a. Numerical problems

In the example in Fig. 3, the teacher wishes to have students discover the conjecture that the sum of the interior angles of a triangle is 180 degrees. The problem is written to highlight that relationship. Typically, these problems focus on only one relationship.

Students are expected to use their pattern finding abilities and numerical skills to generate the desired conjecture.

b. Visual problems

An alternate formulation of the same problem is presented next.

TASK: To explore angle relationships in the following construction.

PROCEDURE:

Draw any triangle ABC.

Through C, draw a segment parallel to AB.

Record a diagram and data.

Repeat this process on other triangles.

State your conjectures below.

In this type of guided exploration problem, the guiding is less explicit. Since the auxiliary line is present the visual representation is more complex. There are more conjectures to be made.

The conjecture that is on the teacher's agenda can be deduced from some of these other conjectures. Thus, in this case, students may conjecture about the equality of the alternate interior angles. These conjectures can help prove the sum of the interior angles conjecture. Thus, with this type of problem, a teacher does not have to depend on students to derive the conjecture that is on their agenda.

In general, to change a numerical guided inquiry problem into a visual guided inquiry problem, one adds an auxiliary line which makes a proof possible. Again the common bond between these two types of guided exploration problems is that the result of the students' exploration is essential to the teacher's curricular agenda.

Application

A different mode of integrating use of the GEOMETRIC SUPPOSER into the classroom suggest that GEOMETRIC SUPPOSER problems are a forum for students to apply and review the material studied in class. For example, after teaching about similar triangles one can give students exercises which ask students to use that knowledge in their work with the GEOMETRIC SUPPOSER. Some teachers might prefer this type of pedagogy to the "Guided Exploration" type because there are larger numbers of possible conjectures with this approach. These problems are open-ended, students are not asked to discover "the" relationship that the teacher seeks. We will illustrate this approach with two types of problems: construction problems and review conjecture problems.

a. Construction problems

A construction problem is a problem that asks students to provide a method to construct a configuration which is described geometrically. Thus, the product that the teacher asks for is not a conjecture, as in the problems described above. Construction problems are application problems because you have to know about the figure that you are trying to build and how to test that you have succeeded. For example, after learning about similar triangles and how to tell when triangles are similar, the exercise shown in Fig. 4 could be given.

In order to solve this problem, students would have to know what similar triangles are and how to check for similarity between triangles. This activity is not a guided exploration problem, because the individual construction methods are not integral to the teacher's agenda. Thus, it is not a central theorem of the course that connecting any random point to the three vertices and finding the midpoints of these segments will identify points that can be joined to make a triangle similar to the original triangle. However, much can be learned from a student's method. Proving these different methods will serve to reinforce the concepts taught in this part of the course.

b. Applying previous knowledge to make conjectures

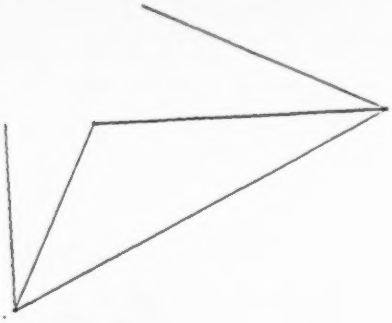
A different way to ask students to review material presented in class is to present a problem whose main expected results have to do with the topic at hand. With this kind of problem, students may come up with other conjectures that are not directly related to the teacher's agenda. As an example, after teaching about similarity, one might give the problem in Fig. 5.

There are many conjectures to be made about the set of figures generated by this construction. One key result is the similarity between the reflected triangle and the triangle which is the connection of the three feet of the altitudes. Again, the teacher's agenda does not depend on the discovery of this similarity relationship. The class can continue comfortably if students do not generate this conjecture, however a teacher whose students did not notice this relationship might conclude that his/her students did not understand the material on similarity.

Going Beyond the Teacher's Knowledge:
Following an Interesting Conjecture

This category describes situations that may occur if teachers are willing to admit to students that they do not know all the geometry that there is to know. In most classrooms, teachers do not give problems to which they do not know the answer. Yet, when using the GEOMETRIC SUPPOSER, students may develop conjectures which the teacher cannot evaluate immediately. Alternatively, students may suggest a way in which the problem can be changed which will present an exploratory situation which the teacher has not thought about before.

For example, in one class, students were given a problem which asked them to find the area of a triangle using three different combinations of bases and heights. The problem asked students first to work in acute triangles and then in obtuse triangles. One pair of students after completing their work on this problem came to their their teacher with the following conjecture. "The angles between the altitudes that go out of the triangle and the nearest side are the same."



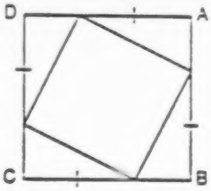
This statement was new to their teacher. Surprised by this conjecture and willing to admit that he had not considered it before, he asked the whole class to investigate these angles and see if they could verify and proof the conjecture of these two students.

In this situation, the teacher did not know the answer to the question that was being investigated and had no particuar content agenda in giving the problem. However, collaboration between teachers and students on this sort of problem may teach students about the process of doing mathematics. Furthermore, the teacher has eloquently made the point that geometry is not a closed subject in which there are no new relationships to discover.

It is also important to note that this approach to exploration, in contrast to the two others, is not characterized by a type of problem that can be isolated from the structure of the problem. Rather the important variable in this case is the teacher's lack of familiarity with the geometrical relationships which reside in this particular construction and a willingness to share the excitement of exploring an unfamiliar problem.

Other examples of this kind of exploratory approach begin with a "What if not?" question about a problem. Such a situation arose during a problem asking students to connect the midpoints of quadrilaterals to create new quadrilaterals (see Fig. 6). The teacher's agenda was to have students apply their knowledge about properties of quadrilaterals and of the mid-line theorem to investigate the kinds of quadrilaterals that are created inside the original quadrilateral and to determine the area and perimeter ratios between these quadrilaterals and the original quadrilateral. Students' exploration began with squares.

The first "What if not?" question was "What if the sides are divided into three or four parts, not two parts?" The area and perimeter ratios were then extended to squares where the sides were subdivided to n segments and then connected in the following way.



Results of this investigation were then generalized to all quadrilaterals which were divided this way. "What if the initial figure is not a square?" The results of this investigation were new and interesting results to the teacher. Finally, an attempt was made to generalize these results to other polygons. These attempts were successful for triangles, but not for other polygons.

Conclusion

The choice of which of these exploratory approaches to use is a personal pedagogical choice. It reflects one's underlying philosophy of teaching and learning and one's view of knowledge (in addition to practical constraints). Hopefully, this outline of approaches will make the type of exploration encouraged by teachers a salient issue and will encourage debate about different uses of the GEOMETRIC SUPPOSER.

Figure 4

Task: To describe methods for drawing a triangle inside ΔABC that is similar to ΔABC such that the two triangles share no points.

Procedure:
- Draw any ΔABC.
- Draw the similar triangle so that it is located anywhere inside ΔABC
- Provide data to verify that your methods work.
- State your conjectures.

Drawings & Data

Conjectures

Figure 5

Task: To explore the figure formed by reflecting the intersection point of the altitudes in each side of a triangle and connecting the three image points.

Procedure:
- Construct an acute ΔABC.
- Draw the three altitudes.
- Label G as their point of intersection.
- Reflect point G in each of the three sides of ΔABC producing points H, I, J.
- Draw ΔDEF and ΔHIJ.
- State your conjectures about the relationships among the points, elements, and triangles.
- Repeat the procedure for other types of triangles.

Drawings & Data

Conjectures

Figure 6

Task: To explore the properties of shapes constructed inside squares.

Procedure:
- Construct any square ABCD.
- Label the midpoints of the sides with points E, F, G, and H.
- Connect points E, F, G, and H.
- Record your drawing and data below.
- State your conjectures.

Drawings & Data

Conjectures

a) EFGH is a ______________________ Why?

b) $\frac{\text{area of ABCD}}{\text{area of EFGH}} =$ ______________________ Why?

c) $\frac{\text{perimeter of ABCD}}{\text{perimeter of EFGH}} =$ ______________________ Why?

ASSESSMENT FOR TEACHING MATHEMATICAL PROBLEM SOLVING: THE COMPUTER LINK

Bethany N. Roditi
Director of Education
Institute for Learning and Development
One Courthouse Lane
Chelmsford, MA 01824

Judith Zorfass
Education Development Center

ABSTRACT

This presentation is based on the "Assessment of Teaching" model (Meltzer & Roditi, 1989) formulated at the Institute for Learning and Development (ILD). This unique process approach to assessment and intervention is guided by current theory and research in child development, psychology, and education. Assessment is conducted for the purpose of understanding how an individual learns best and results in a detailed description of each student's strengths and weaknesses. The learning profile is used to understand each student's learning strategies, to generate classroom teaching techniques, and to develop recommendations for direct strategy instruction.

This presentation will focus on four factors: (1) identification of a child's strengths and weaknesses in developing mathematical problem solving skills and strategies; (2) definition of instructional goals; (3) matching teaching techniques to a student's cognitive and learning profile; and (4) selection and use of computer activities based on the diagnostic information.

The presenter will apply the "Assessment for Teaching" model in terms of a student who was referred for testing due to poor math achievement. We will sketch the student's profile of cognitive strengths and weaknesses with reference to the mathematics domain. We will define instructional goals, select specific teaching methods and materials, and demonstrate how computer activities such as *Collomore Castle* by D.C. Heath offer an effective link between assessment and teaching. This selective integration of technology into the curriculum utilizes the "assessment for teaching" approach and empowers teachers to meet their students' needs.

RECOMMENDATIONS FOR UNDERGRADUATE PROGRAMS IN COMPUTER SCIENCE AND ENGINEERING

A. Joe Turner, Chair
Clemson University
Clemson, SC

ABSTRACT

In the Spring of 1988, ACM and the Computer Society of the IEEE formed a joint task force to develop a new model for undergraduate programs in the discipline of computing. The report from this task force is intended to provide guidelines and recommendations for undergraduate programs named "computer science", "computer science and engineering", and other similar designations, in various academic settings.

The task force has initially concentrated on the core requirements for undergraduate programs, with the objective of defining a common core to support all programs in the discipline. The content of the core has been motivated by considering the minimal body of knowledge and capabilities that every graduate of a computing program should have. The core content includes not only technical subject matter, but also such topics as ethics and professionalism.

Thus far the task force has used the work of the Task Force on the Core of Computer Science (Denning, et al, C. ACM, January 1989) as a basis for defining the core material in the discipline. The none areas of computer science, as defined in the Core Task Force Report, have also been used for the model curriculum. The core content has further been subdivided into topic modules, currently called "knowledge units", that can be combined in various ways for teaching purposes. A collection of principles, concepts, and objectives that occur throughout the curriculum have also been determined, and these are called "recurring themes".

PANELISTS

Allen B. Tucker (Task Force Co-Chair)
Bowdoin College
Brunswick, ME

Keith Barker
University of Connecticut
Storrs, CT

Doris K. Lidtke
Towson State University
Baltimore, MD

SPONSOR: ACM/SIGSCE and IEEE-CS

COMPUTER CONFERENCING FOR THE UNIVERSITY OF THE AIR STUDENTS IN JAPAN

Hiroo Saga
National Institute of Multimedia Education
Japan

Abstract

A telecommunication club was orgainzed for the students of the University of the Air (UA) in order to examine the possibilities and problems of computer conferencing between the long-distance learners. It aimed to allow students a free exchange of messages and also let them learn English in a casual interactive way. Sixty-nine voluntary students were enrolled in the system using a Micro VAX minicomputer with PARTICIPATE II, teleconferencing software. The on-line activity started in September 1987 and lasted 11 months. Of the 69 participants, 51 (74%) could have on-line access to the system, but only 12 (17%) of them sent notes to the topic(s). A total of 443 notes were sent during the period, of which 357 were to "CLUB," the major topic, 40 to "ENGLISH FOR FUN," 20 to "ENGLISH FOR THOUGHT," and 26 to "STUDENT LIFE." The result was discussed in terms of the needs of students, technical problems, the use of English, the role of moderator, and related organizational problems. The potential of giving students a sense of belonging was emphasized.

Introduction

The University of the Air (UA), established in 1985, is a Japanese distance education institute for adults. The UA students continue their independant study mainly through printed textbooks, broadcast television and radio. Like many other institutes of distance education, it is difficult for many students to have a sense of belonging to the university and to their peer students. They feel "the lonliness of the long-distance learner" (Thiagarajan, 1978), and this does affect the attrition rates. As Deutschman and others (1985) state, "a key component of this problem is the communication, or lack of it." This was a major reason why the New York Institute of Technology introduced computer teleconferencing for its American Open University (Richards, et als., 1985). The use of computer conferencing for distance education is now widespread in the United States (Meeks, 1987), and is also in progress in Britain and other European countries (Mason, 1988; MaConnell, 1988; Whiting, 1988; Davies, 1988).

Computer conferencing allows separately located learners to exchange messages without the restriction of time and distance, and hence the National Institute of Multimedia Education, a research institute cooperating with the UA, organized a project team to examine the possibilities and practical problems of applying this medium for the UA students. Since we were not prepared to use it in actual course work, we planned to organize a "telecommunication club" for some voluntary students to freely exchange messages and opinions and therefore English could also be learned in a casual interactive way.

Method and procedure

Host computer and software

A Micro VAX minicomputer with teleconferencing software, PARTICIPATE II, was used as a host system. It received one direct telephone line, four packet communication lines, and three interphone lines with the transmission rate of 300 baud. It could not handle the Shift-JIS Japanese Kanji code, and hence we were limited to use only the English alphabet and Japanese phonetic symbols, Kana.

Preparation for the activity

The project team consisted of five researchers including myself. None of us was fully familiar with the host computer and software, and hence we had to have a certain amount of time for practice while preparing for the project. We first had a consensus as to the general aims of the project and the procedure of collecting students for this activity. It was followed by putting an advertisement in a newsletter asking for students' participation. Out of some eighty respondents, sixty-nine students returned questionnaires for their background data. We then enrolled these 69 voluntary students in our host system in June 1987, and sent them copies of a manual for on-line conferencing. They constitue 0.3% of the total students of the UA. While we were setting up the roles of staff members and the structure of conferencing, the start of the on-line "club" activity was delayed until September because of an accident on the host computer. The conference topics or the particular "rooms" we created in the host computer for this activity are as follows:

1) "CLUB": Here, as a main topic, every member was encouraged to freely exchange any kind of message and opinion. Some discussions of the club itself have also taken place.

2) "ENGLISH FOR FUN": This section was opened as a branch topic of the "CLUB." We intented to provide an enjoyable opportunity to read and write English through conversational interactions.

3) "ENGLISH FOR THOUGHT": This topic also grew out of the "CLUB" to let the members think through various materials using English.

4) "STUDENT LIFE": The aim of this topic was to allow students of the UA to freely exchange opinions and information on course work and other areas of educational experience.

Characteristics of the participants

Among the 69 students, 20% were women and 80% were men. Their ages ranged from 19 to 70, with an average of 36. They lived in different parts of the Kanto area. 80% of them were regular students of the UA. Most of them already had their personal computers, but 10% had neither modems nor acoustic couplers and 35% did not have communication software. 59% of them had prior experience on telecomputing but 7% had no experience with personal computing at all. Their expectations of this club activity were generally very high with 94% of them responding to the question "what is your expectation?" with either "very high" or "high." Their hopes for self efficacy for English typewriting and Japanese word-processing were generally high with only 3, or 7% reponding "has almost no confidence." They felt more confident about handling computers than about using English. In addition to this information, some background data were obtained on their perceptions of different aspects of the instructional system of the UA. This information was later analyzed in relation to the degrees of their on-line participation.

Summary of on-line interaction

Our on-line club activity started in September 1987, and lasted eleven months until July 1988. As an official project, it was ended in April 1988 mainly because of the inactivity of the majority of the staff members and an organizational decision related to it. Since then the on-line activity was maintained by my own voluntary effort.

Actual participation

Among the 69 student members with whom we started the club, those who could have actual on-line access to the host computer were 51 members, or 74%. The remaining 18 members (26%) recorded no access. This figure became smaller as we narrowed our focus to those who joined any one of the conference topics, 46 members (67%), and those who wrote and sent notes to the conference, 12 members (17%) of the original participants. During the last few months of the activity, some 6 or 7 members continued to appear regularly on the conference. While the tendency to have a relatively small number of active participants compared to the number of potential participants is observed in many other networks, we had our own problems which shall be pointed out later.

Amount of access time

Since the start of the activity, 18 members (26%) recorded no access time to the host computer. Other members varied considerably in the amount of their access time, which is shown in Table 1.

Table 1. Amount of Access Time

Hours	Frequency
0	18 (26%)
0.1 - 1.0	12 (17%)
1.1 - 3.0	12 (17%)
3.1 - 5.0	10 (15%)
5.1 - 10.0	9 (13%)
10.1 -	8 (12%)

(as of April 26, 1988)

Number of members in the topics

The first topic, "CLUB," was opened in September 1987 followed by the second one, "STUDENT LIFE." A few weeks after this, "ENGLISH FOR FUN" and "ENGLISH FOR THOUGHT" were branched off the "CLUB." The number of the members who joined these topics is shown in Table 2. The two branch topics for English had fewer members than other topics, which suggests that some members had not accessed the particular note in the main topic which had opened the branch topics. Also, they might have felt some hesitation to join the topics in which only English should have been used. Some students joined more than four topics because there were already several other topics which had been used for different purposes and they could "travel around" them while staying in the host system.

Table 2. Members in the Four Topics

CLUB:	34
ENGLISH-FOR-FUN:	14
ENGLISH-FOR-THOUGHT:	14
STUDENT-LIFE:	35

(as of April 26, 1988)

Number of notes sent to the conference

The total number of notes sent to the topics during the 11 month period was 447 of which 4 notes were deleted by the writers themselves. Then, of the total 443 notes, 357 were to "CLUB," the major topic of the activity, 40 to "ENGLISH FOR FUN," 20 to "ENGLISH FOR THOUGHT," and 26 to "STUDENT LIFE."

Table 3. Monthly Notes to the Topics

	CLUB	E.FUN	E.THT	ST.L.	Total
Sep.	36	2	1	2	41
Oct.	20	6	7	6	39
Nov.	18	1	0	1	20
Dec.	14	3	0	1	18
Jan.	19	2	5	1	27
Feb.	13	4	2	0	19
Mar.	18	9	0	0	27
Apr.	43	3	1	7	54
May.	40	4	1	5	50
Jun.	44	0	1	3	48
Jul.	92	6	2	0	100
Total	357	40	20	26	443

(as of July 31, 1988)

Monthly numbers of notes to each topic are shown in Table 3. The first five notes in "CLUB" were written by me and another staff member to welcome the participants and explain some basic commands. Then the first student appeared on the 6th note followed by other students. The October figure shows that the staff members tried to activate the three new topics. There was a gradual decline of interaction for a few months. A rise was registered in January and in March. A high level was maintainted since April, and in July, the last month, the most active exchange was recorded.

Trend in the contents of on-line exchange

Our on-line exhange started with a welcome note of "hello" and Japanese "konnichiwa," and ended after 11 months. The trend of exchanged contents can be summarized as follows.

1) Stage one (from September to December): The first student sent his note in English as the 6th note in "CLUB." Then gradually, other students began to show up, and also Japanese Kana characters tended to be used more in "CLUB." One participant wrote as his self-introduction that he enjoys observing stars. Another replied that she agrees. Further information was added by a person who had organized an "astronomy club" in his company, and one who had set a telescope in her house, as well as information on open seminars at an astronomical observatory. Other topics included playing jazz, cooking, trips and landscapes, favourite poets, and a discussion of the relationship between nature and human beings. The exchange reflected the change of season from fall to winter, and in December, I sent all the members an on-line Christmas card. "CLUB" was mainly used for this exchange, where I took the role as a moderator.

In "ENGLISH FOR FUN," an American colleague started to activate the topic by giving jokes and quizzes in English, and in "ENGLISH FOR THOUGHT," the same person as a moderator asked the participants to introduce themselves in English. There were some replies to these strategies, but it did not develop further. Another Japanese colleague was to act as a moderator in "STUDENT LIFE," where he was not very active and the role was gradually taken over by me.

2) Stage two (from January to March): A new year started with an exchange of on-line greeting cards in "CLUB." During the winter season, a number of serial notes were sent to "CLUB" by certain member students. They include "One-Point English Lesson," "History of My Computer Use," and "On-line Cooking Class." We also talked about the UA courses, and encouraged each other during the examination period. A short series of rude "scribbling" was also sent by some young students who had been enrolled in the system for another project.

In March, a discussion emerged about how we could continue this on-line activity. The participating members were not fully satisfied with the amount and quality of their exchange. To activate it, they suggested that we might collect new members through advertising. They also expressed sympathy for me, who was at that time the only staff member moderating the conference. In the meantime, it was decided at the host institute to discontinue the project, and I began to plan how to wrap up the activity in a warm and friendly manner.

3) Stage three (from April to June): The discussion about the existence of the club seemed to draw us together, and the exchange since April were at a higher level than before. The serial notes continued to be sent to "CLUB," and various other topics were discussed such as their business trip, favourite noodle retaurants, foreign friends, and international affairs. A note from a young student was also sent confessing his unstable feeling as a UA student, to which I and other senior students gave encouraging comments. In "STUDENT LIFE," I tried to give information about UA programs including the schedule of its television and radio broadcasting.

In May, after I sent a series of notes on closing the club, a number of notes were sent in which the members stressed the great potential of computer conferencing in distance learning and expressed their hope to reopen such a network with a larger body of UA students and faculty members. They also emphasized their lack of mental links to the university, teachers, and peer students, and the merits of computer conferencing to compensate for it.

4) Stage four (July, the last month): In July, the last month, I organized a symposium to evaluate this activity at my institute. Since I announced it to "CLUB," one of the student members participated in the discussion on-line, which was very impressive for many face-to-face participants. Around this time, the UA newsletter for which I wrote about our activity was mailed to all the students. We then could have the first off-line meeting of the club members. By this time, two of the students had already met me in my office. A proposal was sent to "CLUB" from another student to have the meeting. The date and place were arranged on-line with later confirmation by ordinary mail. Fifteen members attended the meeting. The response to this enjoyable event was reflected in the largest number of notes sent in a month. Our activity ended with warm and empathetic notes from the participants saying farewell to the club.

Evaluation and Discussion

Those who could not have access

As there were 18 members who could not have access to the host system, their background information on the pre-activity questionnaires was compared with those who had access. It first showed that no-access members had higher expectations for this activity. The ratio of choosing the item, "expects information exchange on course work," was 67% for the no-access group and 44% for the access group (n=51), and it was 72% for the no-access group and 48% for the access group to the item, "expects questions and answers on course contents." The no-access members had also greater confidence in their use of English as shown on a four-point scale from "high" to "none" (chi square=8.3, df=3, p<.05). Moreover, they had a stronger perception that the learning system of the UA is one-way and encourages little participation of students, as shown on a four-point scale from "agree" to "disagree" (chi square=8.6, df=3, p<.05). These results suggest that if managed properly, this medium can be used effectively to help students in an actual UA context, since those who could not have access had potentially more reasons to use it. My reflection here is that we did not give enough care to these students during the period of activity.

Technical problems and the use of English

We had also some technical problems which limited our activity. The receiving modems or telephone lines were not stable enough at the initial stage. Sometimes, a staff member who was responsible for hardware maintainance forgot to reset the modem switch. The communication speed of 300 baud was slow, especially when the users were waiting for the message after logging-in. The structure of the conferencing software, which uses commands rather than a fixed menu, might have been unfamiliar to the participants, and so I consider that the manual we prepared for them was not a good and accessible one. Another problem is that we could not use the regular Japanese Kanji code. We could only use Kana, Japanese phonetic symbols, which was not very natural, while many other networks in Japan normally use Kanji. The use of English was also rather unique in the Japanese environment of telecommunication. One of our original aims was to give students a chance to learn English in a casual interactive way, but I reflect that to think and write in English without the help of Japanese is not appropriate to many students.

The role of moderator and organizational problems

The most active interaction was made in "CLUB," the major topic. There, participants were freer to exchange different kinds of messages than in other topics. In the two topics for English, the students had to make greater effort to write in English. "STUDENT LIFE," the other topic, did not have a clear characteristic as an independant area. Moreover, I reflect that the moderators made the crucial difference. Computer conferencing is a highly mental medium, and the role of moderators is critical for the amount and quality of on-line exchange. Of the tatal 443 notes, 42% or 188 notes were written by me as one of the moderators. I regret that other two moderators were not active enough to cultivate their topics. Feenberg (1987) empahsizes the moderating role and discusses the need for contextualizing and the "weaving" function. Mason (1988) is observing different moderators on the same course to find various effective styles. From my own experience, they should be good, encouraging, and empathetic writers. They should also be patient, mature, and optimistic, because sometimes they are faced with the rude "scribbling" or emotional outburst of a participant. In this sense, they should provide a model for an etiquette of writing on-line. Moderators need training not only on the technical side of conferencing but also on the human and social aspects of communication.

Another important factor to activate the on-line exchange is the amount and quality of off-line care for participants. It includes preparation of a good manual, initial orientation, correspondence by mail, telephone, and having face-to-face meetings. Our exchange in the last month was significantly higher, possibly due to some of these activities. In this sense, our organizational structure was quite weak. The role of each staff member and the whole plan of the activity were only roughly laid out, and cooperation between the members did not function well. From this, it can be inferred that human relationships and daily communication at an institute can be improved to provide a more condusive context for cooperation.

A new sense of belonging

The most significant result of this activity is that on-line notes from the students were sent in which they realized and analyzed important functions of this medium for supporting separately located learners. They all pointed out that computer conferencing can help them raise a sense of belonging to the University and peer students. One of such notes writes:

> "I have been learning in this changing society for some twenty years since I left school. What I can say from my experience is that the most effective factor to keep on learning is to have peers who learn together, teach and encourage each other. What then about our University of the Air? I must say that there's very little chance to exchange among students and between teachers and students....In this sense, it was a very good experience and a big stimulus for me to have participated in this experiment. In spite of the somewhat small number of participants, our exchange was much closer than what we have at the study center....If, in the near future, this medium is to be introduced into the UA courses and used as a communication tool between the university and students, the instructional quality of this university should improve a great deal....I believe this network could grow to a unique and huge 'think tank' where professional researchers and people working in the real world meet together." (from Toshi, a 41 year old male student and engineer at an airport.)

This note contains a suggestion for functional learning environment (Cohen and Miyake, 1986). Computer conferencing, when human and social conditions surrounding it are fulfilled, has a potential to give social and meaningful context to independant learners who are separately located.

Concluding remarks

In order to reopen the network in an actual situation of the UA, it would be necessary to upgrade the system by introducing the use of the regular Japanese Kanji code and a communication speed of at least 1,200 baud. It is also recommended that we have more receiving telephone lines and access points. More importantly, this activity must include clear objectives, extensive planning and a strong organizational structure with more active participation of staff members in order to support students both on-line and off-line. For this purpose, a good list or guidelines like those proposed by Foster (1985) and Arms (1988) should be prepared. From an administrative viewpoint, it needs to be clarified to what degree this method can be incorporated in or substituted for the existing schooling and correspondence works in the UA institutional condition. To conclude, though it was not a significant success, it provided in microcosm a clear picture of the real issues and potential of computer conferencing as an adjunct to distance education. It is to be highly recommended that care be given to this viable interactive communication medium.

References

Arms, V.M. (1988) Computer conferencing: Models and proposals. *Educational Technology*, March 1988, 43-45.

Cohen, M., and Miyake, N. (1986) A worldwide intercultural network: exploring electronic messaging for instruction. *Instructional Science*, 15, 257-273.

Davies, D. (1988) Computer-supported co-operative learning systems: Interactive group technologies and open learning. *PLET*, 25 (3), 205-215.

Deutschman, W., Kremers, M., Richards, A., Spitzer, M., and Hail, P. (1985) Computer teleconferencing: The next step in distance learning. Kansas State University.

Feenberg, A. (1987) Computer conferencing and the humanities. *Instructional Science*, 16, 169-186.

Foster, D.L. (1985) *Educator's Guide to Conferencing: Using Computers*. Southwest Educational Development Laboratory, Austin; Texas.

Mason, R. (1988) Computer conferencing: a contribution to self-directed learning. *British Journal of Educational Technology*, 19 (1), 28-41.

McConnell, D. (1988) Computer conferencing in teacher in service education: a case study. in D. Harris (ed.) *Education for the New Technologies*. 199-218, Kogan Page, London.

Meeks, B. C. (1987) The quiet revolution: On-line education becomes a real alternative. *BYTE*, February 1987, 183-190.

Richards, A., Deutschman, W., Kremers, M., Spitzer, M., and Hail, P. (1985) On-line at New York Institute of Technology: Computer teleconferencing workshops for faculty. Kansas State University.

Thiagrajan, S. (1978) The loneliness of the long-distance learner. in *Learning Via Telecommunication*. AECT, Washington D.C., 44-45.

Whiting, J. (1988) New perspectives on open and distance learning for adult audiences. in D. Harris (ed.) *Education for the New Technologies*. 67-91, Kogan Page, London.

A NEW GRADUATE TELECOMMUNICATIONS PROGRAM: THE UNIVERSITY AND CORPORATE WORLDS JOIN HANDS

Stuart A. Varden
Associate Professor
Information Systems Department
Pace University
Pleasantville, New York 10570
(914)-741-3527

INTRODUCTION

During 1988, I served as a Visiting Professor with IBM Corporation where I worked on developing, offering and evaluating a graduate-level education program in telecommunications. I was a member of a small curriculum development team that helped develop an integrated credit-bearing sequence of six courses designed to provide IBM internal telecommunications professionals with a solid academic background in telecommunications technology and management.

The effort was an active partnership between IBM and Pace, which has resulted in a highly successful corporate program. Moreover, the six course sequence is now the core of Pace University's newly approved Master of Science in Telecommunications degree program, which is offered through Pace's School of Computer Science and Information Systems.

This paper describes the need for and structure of the program, makes some observations on how the university and corporate perspectives on education differ, and briefly describes the outcome of the first offering of the program which took place in the summer of 1988. Finally, I offer encouragement to universities to seek active partnership with the corporate sector.

THE NEED FOR TELECOMMUNICATIONS EDUCATION

Building and maintaining corporate telecommunication networks has become essential to competitive survival in today's information economy. These networks must provide for the effective and timely communication of voice, data, and visual information within the office, between company sites, and with the outside world.

Moving a large company to be more responsive to the changing needs of the business, however, is most challenging. Many new skills and attitudes are needed to support an orderly change toward doing business a new way. The movement toward network consolidation, integration and connectivity across the company requires that skills be upgraded and that a new perspective on the role of the telecommunications function be adopted. Thus, education must be viewed as instrumental in promoting change.

In response to this need, the internal telecommunications organization within IBM issued a request for proposal (RFP) to the university community in September 1987. The RFP made it clear that IBM was seeking a graduate-level credit-bearing program that was generic in nature. IBM already has developed many IBM product-specific courses for its employees and customers, but had no way of quickly providing its telecommunications professionals with an update to the fast-moving telecommunications field.

THE PROJECT GETS UNDERWAY

Pace University responded to the RFP in October 1987, and was informed that it was the successful bidder in December 1987. The development team of six persons from both IBM and Pace was formed, and work began in earnest in early January.

Pace already was offering a Graduate Certificate Program in Computer Communications and Networks, but several additional courses needed to be developed. It was decided to call the new IBM program the "Graduate Telecommunications Program (GTP)." The overall goal of GTP is to provide IBM telecommunications professionals with career-related graduate-level education through the university. It was understood that GTP participants would be selected for their high potential for professional growth.

GTP participants attend classes at IBM's International Finance, Planning and Administration (IFPA) School located on Pace University's Briarcliff Manor Campus, some thirty miles north of New York City. A brief description of Pace University and the Pace-IFPA relationship would be helpful at this point.

THE IFPA/PACE AFFILIATION

Pace University is an independent institution of higher education founded in 1906. Pace has evolved from an accounting institute to a comprehensive university with approximately 25,000 students on campuses in New York City and Westchester County. The University is chartered by the Regents of the University of the State of New York and accredited by the Middle States Association of Colleges and Schools.

The IFPA School was established in 1980 in White Plains, New York, to provide centralized education for IBM's Finance and Planning staffs. In 1982, the IFPA School moved to its present location in Briarcliff Manor, New York, and joined with Pace University in a forward looking educational endeavor—the IFPA/Pace Program—to assist IFPA in providing superior professional education relevant to IBM's business needs.

Through the affiliation, IFPA and Pace share expertise in curriculum and program design, teaching and learning techniques. A major program outcome of the affiliation has been the approval of many IFPA courses for undergraduate and graduate credit by Pace. The instructional approach and curriculum reflect the best of both corporate and university education. The cooperative merging of resources and perspectives of business and the university has proven to be beneficial to all concerned.

PROGRAM OBJECTIVES

The following broad program objectives were identified and used as guidelines for developing the GTP curriculum:

- To broaden the ranks of middle management and senior professionals with skills in telecommunications technology and management.
- To provide current information and understanding of the telecommunications field including terminology, key concepts, key issues, and trends.
- To provide the analytical tools and skills needed to manage the process of planning, designing, installing and operating telecommunications networks or network nodes in a cost-effective manner.
- To identify sources of information and tools for keeping abreast of key trends in the telecommunications field.
- To develop awareness of the telecommunications environment including governmental agencies, standardizing bodies, professional societies, and a multitude of telecommunications service providers.
- To establish a base of telecommunications knowledge and attitudes to promote further study to include: (a) Continuation of formal education toward an advanced degree in telecommunications, (b) A foundation for lifelong study, (c) An attitude of self-confidence and motivation to keep up with changes in the field after graduating from the program.

PROGRAM OVERVIEW

The Graduate Telecommunications Program emphasizes the role of telecommunications in achieving the operational and strategic goals of organizations. Prominent among these is the recent trend toward physical and logical network consolidation to improve the cost effectiveness of the telecommunications function. Thus, the program takes the view that the knowledge of technical material is of importance to the extent that it contributes to the meeting of business objectives.

The curriculum of the Graduate Telecommunications Program is designed to blend academic subject matter with information based on IBM specific issues and direction. In addition to classroom and laboratory instruction, field experiences and presentations by IBM executives and outside experts provide a balance between theoretical and applied subject matter.

Selected case studies and computer-based simulations are used at several points to help integrate the concepts and disciplines learned in the program by providing a context for complex problem analysis and solving.

COURSE DESCRIPTIONS

The six courses that make up the GTP curriculum are briefly described below.

Data Communications and Networks

This course provides an introduction to data communications as a major component of current telecommunications and computer systems. It progresses from the elements of data, voice and video signals and data transmission to the techniques and components for point-to-point data communication, and an introduction to the operation and design of local and wide area computer networks. Trade-offs arising in the planning and implementation of data communication systems are illustrated by examining several current systems. Topics include:

- Basic Concepts: The role of data communications and trends in telecommunications and computer systems. The fundamental issues of transmission rate, response time, reliability, architecture, and networking. An overview of the OSI layered architecture.
- Signals and Transmission: Analog and digital signals, frequency, bandwidth and data rates, impairments, voice and video signals.
- Transmission Media: The characteristics of wire, coaxial cable, fiber optic media, microwave and satellite transmission; capacity and cost.
- Coding and Modulation: Digital signal formats; amplitude, frequency and phase modulation of carrier signals; modems, the sampling theorem, and analog to digital conversion.
- Data Communication Techniques: Synchronous and asynchronous transmission, error detection schemes, interfacing standards RS-232C, EIA-530, and X.21.
- Data Link Control: Duplex, flow control, error control, delay and capacity; BSC, SDLC, and HDLC protocols.
- Line Sharing: Multipoint, multiplexing, statistical concentration; queueing models for capacity and delay.
- Introduction to Networking: Local and wide area networks, topology, traffic, packet, and circuit switching. Public and private networks.

Computer Networking

This course develops in much greater detail the concepts and services of networks that are introduced in the previous course. The following topics are covered:

- An overview of the ISO Reference Model with discussion focusing on why this model has been proposed, as well as what each layer means and does; overview of SNA, DECnet and X.25.
- The network layer which focuses on virtual circuits, datagrams, congestion controls; examples of the network layer, Arpanet, SNA, DECnet, X.25 and APPC.
- Transport and session layers with a focus on flow control and the role of bridges and gateways.
- Presentation layer with primary topics including network security, encryption, and privacy issues, virtual terminal protocols, and file transport protocols.
- Application layer which focuses on LU 6.2/APPC.
- Network management issues, tools to support network management, e.g. NetView.

Digital Telephony and Switching

This course is an introduction to voice digitization and transmission. The history, development, and future of voice technology are presented with an emphasis on understanding the role of digital telephony in such areas as integrated services digital network (ISDN). The course focuses on the following topics:

- Speech coding and compression, with an emphasis on pulse code modulation, adaptive differential pulse code modulation (ADPCM), and delta modulation.
- Digital telephone transmission, which focuses on synchronization signaling and control, time division multiplexing, T-carrier formats, echo and other impairment, and voice-data integration.
- Digital switching systems, which focuses on space and time division switching, functions, features, and representative switches. Private Branch Exchange (PBX), PBX features and selection, and PBX versus LANs.
- Integrated services digital network (ISDN), features and rationale, and status.

Telecommunications Management

The course is designed to better equip telecommunications managers with the tools, methods and perspective necessary to prepare and critically analyze business situations. Although the ideas may be applied to any business area, examples are drawn from the telecommunications function. The course includes material from the areas of managerial accounting, managerial economics, managerial finance, and quantitative analysis. Topics include:

- Analysis of economic supply and demand.
- Funds flow statement analysis.
- Analysis of financial statements by use of ratios.
- Capital budgeting.
- Lease versus purchase decisions.
- Cost of capital.
- Measurement of investment returns.
- The IBM business case method.

Computer laboratory exercises are included to illustrate the methods presented in class.

Telecommunications Policy and Environment

This course investigates the management problems and public policy issues facing the telecommunications industry as it undergoes structural changes related to deregulation and shifts in government control. A survey of both state and federal government policies and their implications for telecommunications management from a local government to a global perspective are provided.

An integrated analysis of the financial, economic, technological, social and legal aspects of the telecommunications environment are presented through the use of case studies.

Cases in Telecommunications

The issues and tools for planning and implementing telecommunication systems and networks are presented through a major integrative case study. Organizational objectives, legal, financial, and management considerations that contribute to the development, analysis, and implementation of a business case are emphasized.

The integrative course is designed to be the capstone of the program. It is structured to reinforce and put into practice concepts covered in the other five areas. It focuses on the strategies for the effective planning and integration of telecommunications equipment and networks. The course addresses issues such as evaluating vendor proposals, utilizing consulting services, establishing policies and standards, staff training considerations, regulatory legislative and governmental rulings affecting the corporate telecommunications function, and the role of telecommunications in gaining a competitive advantage.

THE FACULTY

IBM was clear from the outset that quality instruction and up-to-date information were top primaries. Although many of the instructors for GTP were drawn from Pace, every effort was made to identify and secure the best available person for each course. As a result, faculty from the University of Colorado, University of New Haven, and several authorities from within IBM and other companies have taught in the program. In addition, a wide variety of outside speakers representing organizations such as AT&T Bell Labs, AT&T, and Sears Communications Network have contributed to the program.

ACADEMIC CREDIT

All GTP participants who successfully complete all courses receive eighteen (18) points of academic credit from Pace University. Two credit options are available: graduate credit for those who already have a bachelor's degree, and undergraduate credit for those who have not completed a bachelor's degree.

Participants who are eligible for graduate credit may apply their GTP credits toward the Master of Science in Telecommunications degree offered by the School of Computer Science and Information Systems. In addition, all successful participants who are eligible for graduate credit are awarded the Graduate Certificate in Telecommunications.

GTP participants who are not eligible for admission to graduate study register for undergraduate credit. An undergraduate certificate is awarded by Pace University to participants who are admitted to undergraduate study and who complete the GTP courses successfully.

All courses are offered on a letter grade basis except for the Cases in Telecommunications course, which is offered on a pass/fail basis. Grades, of course, can be a sensitive matter for anyone, but it is especially true for people who have been away from school for many years. Participants in the GTP are protected by the same Federal regulations as any other student at Pace. Their academic records, including their performance in courses, are not revealed without their prior written consent.

ACADEMIC CALENDAR AND DAILY SCHEDULE

The Graduate Telecommunications Program consists of six weeks of intensive in-residence study spread over a ten week period. The first three weeks are devoted to giving the student a thorough grounding in the technical aspects of telecommunications, telephony, and telecommunications management, while the last three weeks deal mainly with networks, and telecommunications policy and environment. Although participants were not in-residence during weeks 4 through 7, they were given assignments and projects to work on during this period.

Classes normally ran from 8:30 a.m. to 4:30 p.m., Monday through Friday, and Saturday from 8:30 a.m. to 12:00 noon. In addition, speakers, roundtable discussions and field trips are held on selected evenings. Finally, participants are expected to devote several hours each evening to studying and homework assignments.

THE IBM LEARNING ENVIRONMENT AND GTP AUDIENCE

During the development of GTP, several members of the faculty questioned the feasibility and advisability of offering graduate level courses in a condensed form over one or two weeks. It was important to point out that the learning environment and audience for GTP differs significantly from that found in a typical college campus setting. It is true that the intensive nature of the program format does not allow for much out-of-class time for study and assignments. Moreover, there is little time for reflection and "letting things soak in." Nevertheless, there are compensating factors that tend to offset the educational disadvantages of such a concentrated learning situation.

The following observations concerning the IBM learning environment and the GTP audience should help explain why this format does work:

- The participants of the GTP have been selected for their potential for advancement in IBM's telecommunication organization. Their ability and professional background as a group are likely to be superior to that found in a conventional college setting.

- Motivation to learn and work hard is considerably greater than is typical of a college situation. GTP participants are more likely to see a clear and direct connection between their studies and their career development. Moreover, their classmates are also their professional colleagues. Although competition among classmates is not specifically fostered, it is clear that professional pride is a strong motivating factor.

- During the residency period, participants have no other job but to study and learn. Departments that send participants to GTP understand that GTP is a full time educational program, and that participants are unavailable for work-related activities while they are in residence.

- Participants are lodged together at IBM's Corporate Education Center in Thornwood, New York during the residency period. This communal environment is likely to foster a close and mutually supportive working relationship among class members. Informal study groups of about five members are assigned and are expected to meet regularly.

- Except for emergencies or illness, absenteeism and tardiness are virtually zero. Participants are expected to be present and prepared to begin work at the scheduled beginning of class.

- Participants expect to be challenged and can be expected to challenge their instructors. All GTP participants are practicing professionals of IBM's telecommunications community. Interest in the material is high and discussions lively.

- Instructor credibility as an expert in both theory and practice on the topic being taught is a key factor in the success of the program. While it is not expected that the instructor will have IBM specific knowledge, frequent inability to answer more general questions could hurt the credibility of the instructor as well as the program as a whole.

- IBM educational standards call for active class participation. Class sessions include frequent interactions with participants. Alternate activities such as case studies, exercises, role playing, computer lab work or simulations are used to illustrate and reinforce lecture material, and to add variety to the long instructional day.

- It is the IFPA practice that each instructional unit of a course be critiqued by the class participants. This is usually done by means of a one page reaction sheet which is distributed and completed at the end of each day. For new programs such as GTP, a series of interviews with participants also is conducted. This information is instrumental to improving the appropriateness and quality of future offerings.

- The educational facilities and equipment at the IFPA/Pace are superior to those found on the average college campus. IFPA offers comfortable classrooms, breakout rooms for small group work, and a PC/terminal lab for both PC and mainframe computer-based assignments. These resources can be used to advantage in planning alternate educational activities.

In general, there are important differences in the way most corporations view education when compared to the university view. Corporations demand a close tie between what is taught and how it can be applied on the job; universities tend to value conceptual, generic education. Corporations like instructional materials and manner of presentation to be well organized and standardized; universities respect the tradition of academic freedom. It is likely, then, that a basic tension regarding the purpose of education may exist when a corporation and a university work jointly on an educational program. A successful university-corporate program, usually has gone through a period of many long discussions which lead to an appreciation of each other's view. It has been my observation that an educational program benefits from the active "negotiations" that take place.

THE FIRST GTP OFFERING

On June 12th, 1988 a class of twenty two participants began the first offering of GTP with a formal welcome dinner at a nearby restaurant. The composition of the class was quite varied: ages ran from the late twenties to the mid-fifties, fourteen were from the New York City area while the others came from IBM sites around the country, nine were managers and the rest senior professionals, seven already had a master's degree while five had no degree at all, and many different job types were represented. There were data communications specialists, voice oriented people, planners, software developers, marketeers, and financial analysts. The sense of anticipation and excitement ran high as we entered into this new and high profile enterprise. We had worked hard, but had we done enough, and would the program be well received by both the participants and IBM management?

It would take far too long to chronicle all the events, the successes, the mishaps, and the surprises that unfolded over the weeks that followed. Instead, let me summarize the main observations and conclusions that we drew from the experience:

- All participants succeeded in passing all courses, and most did very well.

- Most participants reported working very hard and experiencing considerable fatigue and stress, but the sense of accomplishment in successfully completing the program was very high.

- Many participants experienced difficulties with the mathematical and scientific nature of some of the material. Most participants had been away from formal academic studies for many years, and have had little occasion to use college-level mathematics or science in their jobs.

- The amount of additional support, such as after class review sessions and one-on-one tutoring, was much greater than had been expected.

- A strong sense of common purpose and cohesion developed among the class.

- The traditional "walls" that often exist between data communications and voice people, between software and hardware people, and between applications and systems people, largely disappeared. People who rarely came in contact with one another on the job found that they had to rely on each other to be successful in GTP.

- The class provided very candid and helpful reactions to the program. They had been told from the outset that they had an important role in evaluating the program. Their opinions have had a significant impact in improving the next offering of GTP.

- The GTP faculty has benefited greatly from teaching the GTP audience. They have a much improved insight into how basic telecommunications concepts are being applied in the corporate world. Also, those who had no prior experience teaching in a corporate setting, have exhibited a marked improvement in their degree of organization and ease in front of the classroom. Thus, involvement with GTP has proven to be an outstanding faculty development experience for several of our faculty.

- About a third of the class has decided to continue toward the Master of Science in Telecommunications degree at Pace.

But has GTP made a difference in the long run? Will GTP help IBM be more successful in moving ahead with its plans to consolidate and integrate its internal networks? This is perhaps the ultimate question for any corporate education program.

IBM Education has recently introduced an approach to evaluating its education programs which goes beyond seeking student reactions and testing what they have learned through an examination. IBM is now beginning to conduct post-program interviews with participants and their managers to learn if on-the-job performance and attitudes have changed as a result of the program. Although formal interviews with the participants of the first GTP class have not yet been conducted, the informal comments are that GTP is proving to be even more valuable than they has imagined. One graduate bemoans not having had such a program five years earlier. "I now see that I could have done a much better job for the company" he points out.

Meanwhile, GTP II is currently underway, and GTP III and GTP IV are scheduled for the first half of 1989.

SPREADSHEETS AND CRITICAL THINKING

Tony Jongejan
School of Education
Western Washington University
Bellingham, WA

In recent years there have been a number of studies and reports that recommend an increased emphasis on the teaching of critical thinking. Critical thinking is usually tied to the higher levels of Bloom's Taxonomy of the Cognitive Domain and thus, working with student's critical thinking ability usually entails emphasizing higher order thinking skills. Bloom presents a hierarchy [(1) knowledge, (2) comprehension, (3) application, (4) analysis, (5) synthesis, and (6) evaluation] of learning which recognizes that different skills are required at different levels of thinking.

The lower level of the hierarchy consists of facts and skills that define a discipline. The goal of teaching involves students improving their recall, speed, and accuracy with the result that students can solve simple, low-level problems of the discipline. At the higher level of the hierarchy, concepts such as analysis, synthesis, interpretation, evaluation, inquiry, reasoning, and problem solving are emphasized. The goal of teaching involves providing students with opportunities to develop these abilities both within the discipline and across disciplines. Work with higher order thinking skills is necessary for all students but, since the lower level facts and skills are the building blocks one uses when doing critical thinking, work with these must not be neglected.

Computers (and appropriate software) are one vehicle for providing students with opportunities to develop their higher order thinking skills. The important issue here is the selection of the appropriate software for the skill being emphasized. Thus, if one is working on the student's ability at visual thinking, an educational game such as *Building Perspective* by Sunburst would be an appropriate choice. To help with this software selection, a hierarchy of higher order thinking skills software was developed.

Levels of Higher Order Thinking Skills Software

1. Tutorials provide opportunities to learn Higher Order Thinking Skills;
2. Instructional Games provide opportunities to develop specific Higher Order Thinking Skills;
3. Simulations provide opportunities to develop general Higher Order Thinking Skills;
4. Construction Sets provide opportunities to manipulate objects in a specific environment;
5. Application Software (e.g., Data Bases and Spreadsheets) provides opportunities to create and use tools;
6. Computer Languages provide opportunities to program the solution to problems.

Although the ease of use increases as one moves up this hierarchy, the flexibility, power, amount of instruction, and the payoff for critical thinking increase.

Spreadsheets provide students a tool to uses in problem solving. But they are also a powerful tool in the hands of an effective teacher. Specifically, teachers can develop templates that provide students with many of the opportunities listed in the hierarchy of software. This session will discuss some of the issues mentioned above, give examples of software that fits in the various categories and demonstrate examples of spreadsheet templates that also can be used. Finally, the issue of transfer of higher order thinking skills and techniques for increasing this transfer will be addressed.

SOFTWARE DESIGN FOR LEARNING: ON LEARNING MATHEMATICS THROUGH INSTRUCTIONAL SOFTWARE DESIGN

Idit Harel
The Learning Research Group, Media Laboratory
Massachusetts Institute of Technology
20 Ames Street, Rm. 310
Cambridge, MA. 02139

Abstract

This paper describes a four-month-long experiment in computer-based learning called Instructional Software Design (ISD) and presents aspects of the evaluation. The participant ISD class, comprised of fourth-grade children at a Boston public school, learned about programming and fractions while developing instructional software, in LogoWriter, for third-graders. Evaluation was quantitative and comparative as well as qualitative. Two control classes were selected: one class had equivalent exposure to Logo, but no exposure to ISD; the other used Logo only once a week in a "computer laboratory." All three classes followed the regular mathematics curriculum, including a two-month unit on fractions which coincided with the ISD project. Pre- and post-tests were administered to the experimental and control groups; in addition, the experimental group was carefully observed and interviewed throughout. The evaluation revealed greater mastery of Logo and fractions by the experimental class than for either control class and greater acquisition of metacognitive skills. In the course of discussion, the ISD approach of using programming as a tool for reformulating knowledge is compared to other approaches to learning Logo, notably teaching programming per se in isolation from a content domain. In the final section, suggestions are given for future research and development.

Description of ISD: The Experimental Treatment

The Instructional Software Design Project (ISD) involved a class of seventeen fourth-graders at all levels of mathematics achievements in the task of developing instructional software about fractions for third-graders. As they learned about fractions themselves, both in their regular math curriculum and informally in ISD, each child planned, designed, and implemented software in Logo to explain fractions concepts to others. They concurently learned about Logo and fractions in the context of thinking about teaching others.

Instructional software design is a complex, active, and extended enterprise. It requires both understanding and creation. It requires skill in programming as well as knowledge of the content and a feel for instructional design. The designer is involved in learning and in making concepts explicit.

ISD was open-ended but was structured around a series of activities. Each working day, for five to seven minutes, children wrote out their plans and drew their designs in individual "Designer's Notebooks."[1] They then worked at individual computers for 45 to 55 minutes. At the computer, the children implemented their plans and designs, created new ones, and revised old ones. When they wished, they were allowed to work with their friends, to help each other, or to walk around to see what other children were doing. When the computer time was over, children individually saved daily files on a special diskettes and returned to the classroom. In their Designer's Notebooks, they would then record reflections on the problems and changes made that day, sometimes adding plans and designs for the next. The children had full freedom in choosing which concepts they wanted to teach, how to design their screens, what the lesson sequence should be, and what testing to include, if any. The only two requirements were: 1) that they write in their Designer's Notebooks before and after each working session; and 2) that they spend a specific amount of time at the computer each day.[2]

Children used their Designer's Notebooks for designing and drawing screens, story-boarding, planning, and reflecting on changes and problems. The writing in the Designer's Notebook was of key importance for the project as well as for for children's learning and cognitive development. We did not tell the children what or how much to write, what or how to plan or draw, or how to reflect on their work or make changes. We did, however, explain to them that they, like professional software designers, might find it useful to keep track of their ideas and changes, since this could aid their concentration and help them to hold on to good ideas from one day to the next, thus contributing to their finished product.

Writing in Designer's Notebooks was clearly a habit which would take a little time for children to acquire. By ten days into the project, however, the Designer's Notebooks really seemed to take hold and become a personal learning tool. Children used them as a source of ideas and got in the habit of moving back and forth between plans, drawings, implementation, and revisions.

The teacher and researcher collaborated and actively participated in all the children's software design and programming sessions. We would walk around the children, and sometimes sit down next to them to look at their programs and give them help if they needed it. We often discussed design and programming issues with the children individually in this way. In general, we had no specific plans for the ISD's sequence or for our discussions. We initiated them at times relevant to the children's work or problems or at the children's request.

In addition to this work with individual students, several short "focus sessions" about software design, Logo programming, and representations of fractions were held in the classroom. These sessions lasted 5 to15 minutes each. In the first one, we briefly discussed the concepts of instructional design and educational software, and showed them designs, plans, and flowcharts from past instructional design projects of the researcher. We also circulated Lammer's (1987) book, Programmers At Work, and asked the children to examine the notes, pieces of programs, and designs of professional hardware and software designers and programmers (including those of designers of the Macintosh, PacMan, Lotus 1-2-3, and others). In this first session, the children also received their personal diskettes and Designer's Notebooks, and we discussed the ways in which they could be used during the project. During the other focus sessions we concentrated on issues such as the difficulties of specific fraction concepts and the children's ideas on how they might be explained, represented, or taught.3 Other Focus Sessions concentrated on specific Logo programming skills.[4] In the end, all children each created a piece of instructional software about fractions, and a personal software-design portfolio. The individual portfolios consisted of daily sets of written plans, designs, pieces of Logo code, and written reflections on problems and changes.

ISD as a New Approach for Learning Logo

Elementary-school children's difficulties learning Logo have been well documented (Pea & Kurland, 1984; Kurland & Pea 1983; Carver, 1986, 1987; Heller, 1986; Perkins et al. 1985; Salomon & Perkins, 1986; Clements, 1984). However, although programming (in Logo and Basic) has been found to contain many difficult constructs, it will probably remain in the curriculum of many schools if only for the reason that many educators and parents believe that computation is an important piece of knowledge, and that computer programming is an important skill for all children in a technological society.

Many claims have been made about what knowing, learning, and understanding Logo programming means to young children, and about cognitive changes learning to program in Logo is expected to produce (Papert, 1980; Feurzeig, Horwitz, and Nickerson, 1981; Pea & Kurland, 1984).

The very first and most general claim about what learning to program in Logo means was provided by Papert (1980), who stressed the reformulation of knowledge--rethinking and relearning "old" domains to the point that essentially new domains of knowledge would emerge. For example, Turtle Geometry represented the emergence of a dynamic computational and locally-described geometry as against the older static paper-and-pencil and globally-described geometry. Thus, early in its history, Logo was proposed as a learning tool for learning other things (which involved areas of mathematics, physics, astronomy, robotics, and cybernetics), as a functional tool for acquiring and manipulating other kinds of knowledge, as an "educational philosophy," or as a unique "learning experience," rather than as a set of concepts and skills to be mastered in isolation for their own sake.

Papert also suggested that once Logo was widely used in society and integrated into many school subjects (like word processors today), the particular learning experiences more or less inherent in Logo would bring major changes in learners' attitudes towards learning in general, in their self-images as learners, in their self-esteem, and in their cognitive development.[5] Through Logo programming, children could experience various kinds of thinking, among them creative and impressionistic thinking, as well as the rigorous thinking and precise expression needed to make computers function. Eventually, children could learn how to suit the style of their thinking to the occasion. Programming would also provide an introduction to the use of heuristics (e.g., planning, finding a related problem, exploring alternative solutions, or solving the problem by decomposing it into parts), which they then could apply in other learning situations. Developing a positive and skilled approach to the debugging of errors was particularly stressed as was modularity, the idea of using small, reusable procedures as building blocks of larger programs or solutions. In the process of programming, children could also learn that there is rarely a single "best" way to do something, but different ways that have comparative costs and benefits with respect to specific goals.

Of course, to get to these larger effects, it is necessary to learn to program in Logo. Thus, specific programming concepts had to be acquired, including use of procedures and sub-procedures, variables, conditionals, iteration and recursion. These concepts, it was claimed, were not just means to a larger end, but were powerful concepts in their own right.

A review of the major research on children, programming, and cognitive development, reveals an emphasis on issues such as children's difficulties in learning to program, in debugging and modularizing programs, or on how to use inputs, variables, conditionals, recursion, lists, as well as on extent of "transfer" to some other skills (e.g., Carver, 1987; and Pea and Kurland, 1984). But the ideas that lay behind Logo's early development have received far less attention, although several attempts have been made to conduct more holistic and constructivist studies about young children and Logo (e.g., Lawler, 1985; Papert et al., 1979; Solomon, 1986; Turkle, 1984; Watt, 1979; Weir, 1986).

The general response to findings that children had difficulty acquiring Logo skills and concepts or failed to transfer Logo skills to other domains, has been to create "better" instructional techniques. More often than not, these have involved an imposition of traditional teaching methods onto Logo, in which Logo was treated as programming per se, isolated from any other subject area (e.g., Carver, 1987; Perkins et al, 1985; or several articles in Pea and Sheingold, Eds., 1987). Many of the pessimistic interpretations of the findings in the research on children's learning of Logo programming were a result of treating Logo as an object of knowledge in itself rather than as a tool for acquiring other learning, and focusing on short-term incremental changes in the learner, rather than viewing Logo and computers as a trigger for broad changes in a learning culture, which might only be observed over a long period of time. While these issues cannot be addressed in more depth here, they have been considered elsewhere (Papert, 1985).

In the ISD study, Logo was used in a broad and complex situation, was integrated into a larger context of software design and development, and served as a tool for accomplishing other learning. Thus Logo learning was not trivialized in the course of setting to study it. At the same time, Logo activity was part of an existing learning community, and reflected the everyday concerns of a realistic educational setting.

ISD as a New Approach for Learning About Rational Numbers and Their Representation

Elementary-school children's difficulties in learning fractions and understanding their representations have also been well documented (Carpenter et al., 1976; Lesh & Landau, 1983; Behr et al, 1983; Peck & Jencks, 1981; ETC, 1985; Post et al., 1985; or Janvier, 1987). Unlike whole numbers, which children largely come to grasp informally and intuitively out of school, learning the rational-number system is confined almost ecxlusively to school. Because rational-number concepts and algorithms are so difficult for so many pupils, they figure prominently in the curriculum each year from the second grade on. Even so, several national assessments of children's mathematical achievements have found that children's performance on fraction-ordering and computation, for example, was low and accompanied by little understanding (Tierney, 1987; Post et al., 1985).

Fractions are ideal tools for learning about number systems and representational systems in mathematics. The understanding of the rational-number representational system is a privileged piece of knowledge among the other pieces of rational-number knowledge. Representations form part of the deep structure of rational-number knowledge, wheras algorithms are the surface structure (e.g., Janvier, 1987). One of the goals of ISD was to involve children in exploring and learning the system of representations of fractions intensively and in a special way, and then assess their knowledge of basic rational-number concepts and algorithms.

There is a diversity of knowledge about rational numbers, including 1) the subconstructs (e.g., ratio, part-whole, operators, fractions, decimals, or percentages), and 2) the representations for each subconstruct and for the whole rational-number system (e.g., words, mathematical symbols, pictures, or real-life situations). When the child "visits" a particular rational-number subconstruct, he or she will be sensitive to the properties and characteristics of that subconstruct. Several of these subconstructs will be more or less intuitively accessible, and some children might be more familiar with, and think more easily "in the style of" one subconstruct (e.g., fractions), while at the same time they may be less familiar or feel uncomfortable thinking in the style of another (e.g., decimals). In general, the relations between subconstructs are poorly organized and unevenly formalized (e.g., Behr et al.1983).

The whole rational-number system and its subconstructs derive some meaning from each other, and multiple representations are not just alternative means of understanding, but are viewed here as the deep structure of rational-number knowledge. Therefore, it was important to help children move easily from one subconstruct to another, to connect and differentiate between each subconstruct's characteristics and properties, and to express the same ideas using several representations such as sections of circles or rectangles, words, money, food, or time. These kinds of connections are facilitated by understanding the translation processes between rational-number representations.

A major focus in previous assessments of children's development of rational-number concepts has been the role of manipulative aids including Pattern Blocks (e.g., Harrison, 1972), Fraction Bars (e.g., Bennett, 1981), Cuisenaire Rods (e.g., Davidson, 1977), and Number-Lines (ETC 1985), etc. In these experiments, the materials were used to facilitate the acquisition of rational-number concepts and representations, as the child's understanding moved from the concrete to the abstract (Davidson, 1977). The approach in ISD was very different. The psychological analysis done by Lesh et al. (1983) , for example, showed that manipulatives were just one form of presentation in the large representational system, and that the other modes of representation (the symbolic, written, or real-life situations) also played a very important and variable role for different thinking styles, and in children's acquisition and use of these concepts. Different materials and activities were found to be useful for making models of different situations, and no single manipulative aid was found to be the "best" for all children, for all rational-number situations, or for translating fractional representations (Behr et al., 1983).

Using Logo to Represent and Combine Rational-Number Representations

Consistent with Lesh's psychlogical analysis of representations, the ISD project did not focus on children's working with one subconstruct or any most powerful representation (e.g., as in the Number-Lines or Pattern-Blocks curricula). Instead, ISD provided children with an environment in which they could work and explore relations between several representational modes (e.g., pictures and symbols), combining the different rational-number subconstructs (e.g., connecting 50% and 1/2), and translating between several representational modes (e.g., designing a screen that combined both graphical and written representations for the fraction 1/2, and the decimal 0.5).

For example, in Logo a child can program a simple picture of a circle region divided into fourths and, using different flashing colors, shade in two of these fourths and have them blink on and off in order to show a representation of two-fourths. The child can add the written words "two-fourths," which is translating pictures into written words. She can add another picture--a large round clock with an animation of the clock's big hand moving slowly from the number 12 to the number 3--and write, "this is one-fourth of an hour," then move the hand from number 3 to the number 6 and write, "this is another fourth of an hour"; or write on the screen, "one-fourth of an hour is 15 minutes, two-fourths are 30 minutes" (this is a translation of the pictorial representation of time or clocks into words, but it is also a representation of a real-life situation and its translation into fractions).

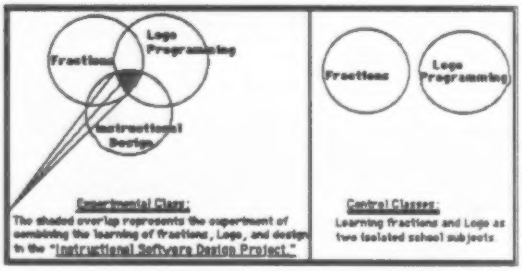
Another example, taken as these all are from the children's actual projects, is to program a picture showing a one-dollar bill with four quarters underneath. Two of these quarters be highlighted in different colors, and be animated to "walk" around the screen and "sit" beside the written words "two-fourths of one dollar." Another approach would be to compose a musical tune, then play a half of it, a fourth of it and so on. We can imagine a variety of representations, from pizzas to gears, from musical rhythms to body movements. As one of the children put it, "Fractions can be put on anything!"

The ISD children used Logo to make their own representations of rational-number concepts for teaching. In so doing, they were trying to make the representations that would serve as good pedagogical aids for other children. By becoming designers of instructional software, the children gained distance and perspective in two senses. In the first place, they were dealing not with the representations themselves, but with a Logo representation of the representations. Moving between representations was subordinated to programming good examples of representations. Secondly, the children programmed, not for themselves, but for others. They had to step outside and think about the other children's reactions. This distance and perspective provided children with control of the process of learning and moving between representations, contrasts with being put through the paces of an externally-conceived sequence of learning.

In summary, ISD recast fractions learning in essentially two ways: 1) it emphasized more involvement with the deep structure (representations) over the surface structure (algorithms) of rational-number knowledge; and 2) it made fractions learning instrumental to a larger intellectual and social goal, that is, having children explain what they think and learn in an interactive lesson for younger children.

The Evaluation of ISD

The evaluation was designed to examine how children who learned fractions and Logo through the ISD project differed from children who learned fractions and Logo through other pedagogical methods. The following diagram sketches the differences between the experimental and the other pedagogical approaches that were assessed for this evaluation:



Experimental Class: The shaded overlap represents the experiment of combining the learning of fractions, Logo, and design in the "Instructional Software Design Project."

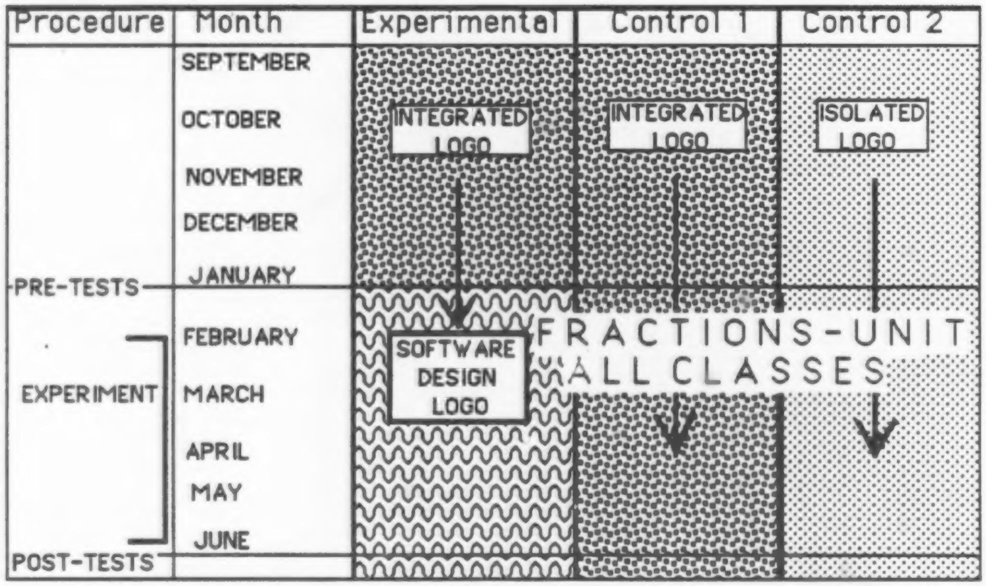
Control Classes: Learning fractions and Logo as two isolated school subjects

Three fourth-grade classes from the same inner-city public school in Boston were selected for this evaluation.[6] The experimental class, from Project Headlight,[7] (N=17) was involved in the ISD project. Control Class 1, or C1 (N=18), studied fractions only in their regular math curriculum and programmed in Logo as part of Project Headlight. Control Class 2, or C2 (N=16), studied fractions in their regular math curriculum, was not part of Headlight, and programmed only once a week in the school's "computer laboratory." All pupils had been tested by their teachers for mathematics level at the beginning of the year, and had been divided into low-, medium- and high-math groups in the following way.

The selected classes divided into their math levels:

	Experimental	Control 1	Control 2
Low Math	N = 4	N = 5	N = 5
Medium Math	N = 5	N = 6	N = 5
High Math	N = 8	N = 7	N = 6
Total in Class	N = 17	N = 18	N = 16

In January 1987, all three classes were pre-tested on specific skills and concepts in fractions and Logo. The pre-test results were consistent with the prior division into mathematics levels. In other words, all the low-math children scored low in the pre-tests and all high-math children scored high. After the pre-tests were completed, the four-month ISD experiment was conducted with one of these classes. All 51 pupils were tested again in June (when ISD ended) on their knowledge of fractions and Logo. This evaluation procedure is represented in the following diagram, with a brief recapitulation of the differences between the three classes.



Fractions Unit means that all teaching of fractions, for all three classes, was conducted for two months in math lessons and following the traditional curriculum. The experimental class was not provided with additional didactic instruction on fractions during the project.
Isolated Logo (in C2) means that a) the children programmed once a week in the school's computer laboratory; b) they followed a conventional approach to Logo programming, using worksheets, exercises, etc., as part of a computer-literacy unit; c) the programming activities resulted in children's completion of short programming exercises, but not in creative products.
Integrated Logo (in C1) means that a) the children programmed every day; b) they integrated the Logo projects into various topics of the curriculum; c) the programming activities resulted in children's completion of various small-scale creative programming projects, each lasting approximately 1-3 weeks.
ISD Logo in the experimental class a) the children programmed every day; b) they integrated Logo into the curriculum, but in the context of designing and programming instructional software for teaching about fractions; c) each child worked on the project for four months, and the programming activities resulted in a long and complex product.

Evaluation Objectives and Questions

Many research questions could have been raised concerning the ISD experiment since it involved so many variables within an integrated and complex pedagogical situation. However, for the purpose of this study, the objectives and questions were narrowed down into two main sets of assessments: 1) an assessment of the experimental children's knowledge of basic fraction concepts; and 2) an assessment of the experimental children's knowledge of Logo programming concepts and skills. The experimental treatment interlaced the experimental children's learning of fractions and Logo with the designing and programming of instructional software. Since the experimental children had equivalent, though differently-styled, exposure to Logo as C1 (i.e., both classes were part of Project Headlight), it was an open question as to whether participation in ISD would result in greater or even equivalent Logo knowledge. One naturally expected both of these groups to exceed C2 in this area. With respect to fractions learning, the experimental group had additional (but not formal) exposure to fractions concepts through ISD, so that improved performance of the experimental class was expected, but the assessment sought to determine whether this was true and, if so, what the nature of the improvement was. As will be seen in the next section, the assessment uncovered some surprising results, more finely-textured than these general surmises.

Using the set of pre-tests, the differences between the experimental and the control children's knowledge of fractions and Logo before the experiment began were investigated; a base line was established, and no significant differences were found. Four months later, using the set of post-tests, the ways in which these children differed in their knowledge and understanding of fractions and Logo after the experiment was ended were investigated in detail.

One part of the post-evaluation aimed at assessing how much better the children from the experimental class were at understanding fractions. Within this domain, emphasis was placed on their ability to translate between various modes of fractional representations. This aspect has been shown to be a crucial part of rational-number knowledge, and particularly difficult for young children (e.g., Lesh et al, 1983; Behr et al., 1983; and others). But standard tests were also used, which concentrated on children's use of algorithms.

The other objectives of the evaluation gave specific attention to the children's knowledge of Logo programming after the experiment. For this purpose, the evaluation investigated the children's knowledge, use, and understanding of Logo programming commands, instructions, and operations. More specifically, it assessed whether the children from the experimental class knew and understood more programming commands and operations such as REPEAT, IFELSE, SETPOS, variables, and inputs in their projects, and became better at these skills than the children in the two control classes. The evaluation also investigated whether the experimental children could understand, implement, debug, transform, optimize, and modify someone else's programming code better than the children from the control classes. Finally, the evaluation assessed whether the experimental children were able to construct Logo routines for someone else's design or picture and were better at this than the children in the two control classes. In addition, given the breadth of the learning experience and the mixed methodology of assessments--including the extensive case studies of several children (i.e., examination of the children's progress, Designer's Notebooks, finished products, interviews with participants during and following completion of the project), as well as the objective pre- and post-tests--it was also possible to draw inferences regarding acquisition of metacognitive skills, and to trace in detail the microgenesis of Logo and fractions skills and concepts, exploring different approaches taken by children with different personal and learning styles (Harel, 1988a).

Procedure and Instruments Used for the Evaluation

A series of tests on fractions and Logo programming was administered to the experimental and the control classes before and after the experimental project. A short description of each of the testing items is given below. There were two versions for most items, one for the pre-tests and one for the post-tests. However, the pre- and post-tests were generally identical in conceptual scope. The post-tests were usually longer and more demanding with difficult questions added on. In most cases, pre-questions also appeared in the post-tests.

The Rational-Number Concepts Paper-&-Pencil Test. A set of 43 multiple-choice questions in the pre-test, and 65 questions in the post-test, including written language, written symbols, and several types of pictorial representations was used to assess the childrens' understanding of rational-number concepts before and after the experiment. This paper-and-pencil test was taken from a collection of tests developed for the Rational-Number (RN) Project by Behr, Lesh, & Post (Lesh & Hamilton, 1981; Lesh, Landau, & Hamilton, 1983). The researchers of the RN Project discovered and documented many of the problems young children encounter when translating these representations of rational numbers. The researchers of the RN Project had developed these tests for investigating the nature of children's ideas about rational numbers in grades 2 through 8. The particular questions selected for this evaluation were geared towards exploring children's ability to translate within and between representational modes (written language, written symbols, pictures) on sets of structurally-related tasks. All tasks in the selected test assess children's basic concepts of fractions and ratios; they also assess children's ability to translate representations, ordering, equivalent fractional forms, and simple proportions. The test was administered to each class separately but on the same day. Children were given 60 minutes to complete their tests.

Boston Public Schools Math Test, Level 4. The Math Level 4 Curriculum Referenced Test, which is given to all fourth-graders in the City of Boston at the end of each year, was used in the present study as a post-test only. The test included 40 multiple-choice questions, of which 15 were about fractions. Children had up to 60 minutes to complete this test.

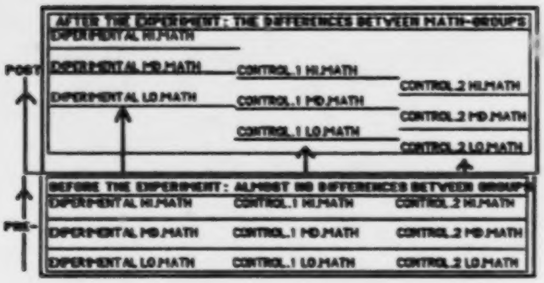
Logo Pencil-&-Paper Test. This test included two open-ended tasks and three well-defined tasks. It was designed by the researcher to assess: a) the children's understanding and use of Logo instructions and commands; b) their ability to follow and execute a given code; and c) their ability to "chunk" a given program using sub-procedures, REPEAT commands, and variables. The post-test included all of the pre-test with additional, more difficult sub-tasks. These tasks require that children: a) generate Logo code for a given picture; b) follow and draw the results expected to be produced by a given code; and c) manipulate the inputs for circles, half-circles, rectangles, and squares. Each of these Pencil-&-Paper Logo Pre- and Post-tests lasted 45-60 minutes.

Logo Computer Test. This test was designed by the researcher to assess the children's knowledge and ability to use Logo while working at the computer. This task required that they debug a program given to them on line by analyzing the picture generated by a buggy program. The children had to modify the debugged program according to the researcher's requests ("make it modular..." or "try to use variables now..." etc.). The post-test was the same as the pre-test, using a slightly more complex picture. However, the underlying structure and the requirements of the pre- and post-Logo Computer Task Tests were identical. Each of the tests lasted 45-60 minutes.

Results

Here we present selected results from the evaluation. Since no significant differences between the three classes were found in the pre-tests, this section presents only some of the results from the post-tests, as well as some general results from interviews and observation.

The overall trends in post-tests, among the experimental and two groups of control children, at all three math levels, are sketched in the following diagram:



In general, the 17 children of the experimental class did better than the other 34 children on all post-tests (fractions and Logo). The experimental children from the high-math group were consistently better than all the other children. The experimental children from the medium-math group, in many cases, were better or the same as the high-math children from the two control classes, and consistently better than the control children from the medium or low-math groups. As for the low-math children from the experimental class, they were usually better than or the same as the medium-math children of the two control classes, and consistently better than the control children in the low-math group. The experimental children showed significant gains over the control children in acquisition of metacognitive skills, and expressed this mastery in post-tests, in interviews, and in their observed progress throughout the project.

Summary of the Results from the LOGO POST-TESTS

Examples from the Logo Pencil-&-Paper Test: This test included seven questions which covered many aspects of the children's knowledge of Logo.[8] The following table summarizes the questions and the skills assessed.

Ques:	Skills Assessed:
Q_1	basic knowledge and understanding of the Logo language commands, and their uses.
Q_2	a classification task of the Logo commands; examined what relationships children perceived between different Logo commands and whether they thoughtabout them as general classes or families, or as loose functional and more task-specific.
Q_3	read a linear Logo code, understand its flow of control, and draw the picture accordingly.
Q_4	modularize and optimize the code given to them in Question 3.
Q_5	understanding of inputs.
Q_6	transform several given procedures according to specified requests.
Q_7	more deeply investigated the children's ability to read and understand short but quite confusing Logo routines and execute them through drawing.

For example, in question 1, the children were asked: "Please list all the Logo instructions and commands that you know and use--in column A; then, write an explanation and give an example for each one--in column B."

The results for this question were divided into two major groups of findings. The first are the simple findings that relate to how many instructions and commands each child actually listed. The second relate to the children's understanding of the meaning and functions of these commands and instructions in the Logo language. The following table represents the differences between the children in terms of how many Logo commands, operations, function keys, control keys, etc., they listed in the post-test.

	EXPER.	CTR 1	CTR 2
HI	32	16	8
MD	24	9	8
LO	21	11	7
AVE	25.6	12.0	8.3

Logo P&P Test, Question No.1

The number in each slot of this table shows the average number of commands & instructions each child's listed and explained in each math-level and class. Each child from the high-math experimental class, for example, listed and explained an average of 32 commands and instructions. The advantages of the experimental children of all math groups over the children from the two control classes become clear from examining this table.

The children were also evaluated on the quality of their definitions and examples for each of the items they had listed. In C_1 no one was evaluated as "VG" (Very Good), whereas in the experimental class, three students who wrote over 40 commands and instructions and gave a very good example and definition of each, were evaluated as "VG." No one was evaluated "L" (low) or "VL" (very low) in the experimental class. However, four children in C_2 were evaluated as "L" since they listed less than five commands and instructions and did not provide examples or definitions for all or most of those.

To take another example, question 3 focused on the children's ability to analyze given programming code and "execute" it on paper. A long, linear Logo code composed of short strips of Logo primitives was given to the children, and the children were asked to draw the graphics. This task required that children read the given linear code, comprehend it, understand its flow of control, build a mental model of what the computer would do when each of the lines in this program was executed, and draw the picture accordingly, step by step.

Many researchers in the field of programming distinguish between writing a linear program and a modular program. These researchers consider a linear program as one which emphasizes the generating of effects without any consideration and understanding of the inner structure of the code (e.g., Kurland et al. 1987; Carver, 1987; Soloway, 1984; and others); on the other hand, a modular program emphasizes elegant and efficient programming, and is accomplanied, they claim, by a higher-level of understanding of programming in general, and of the programming language characteristics in particular (ibid.). These researchers also claim that children usually find it quite difficult to write modular programs, and that children should learn very early on in their process of programming how to write them, since, "programs that consist of long strips of Logo primitives [i.e. linear programs]...are nearly impossible to read, modify, or debug, even for students who have written them" (Kurland et al., 1987).

The results from question 3 show that the more knowledgeable children and who, during their process of learning to program, had written linear as well as modular programs, were also better at understanding and correctly executing this confusing linear program. Whereas the children who only knew how to write linear programs were not able to solve this problem accurately. We should note that experimental children often introduced structure into their programs (i.e., subprocedures and functional naming) only after a long period of purely linear programming. They introduced it when they themselves saw the need; it was not imposed on them from the outside.

Another interesting aspect of these results came to view in the "number of trials" category. Many of the experimental children tried more than once to draw the picture and finally found the right solution; but the children in the control classes who had gotten it wrong in their first trial were apparently not motivated or determined to try again or to find the right solution. Many of them simply wrote "I don't know how to do it," and went on to the next task on the test.

Question 4 assessed the children's abilities to modularize and optimize procedures. The children were asked: "After you finish your drawing (from question 3), try to simplify this program, or 'clean it.' In other words, think about other ways to write a Logo program that will produce or make the same drawing. Are there Logo commands that will make this program shorter? (Also, think about creating sub-procedures)."

In order to answer this question, the children had to make the program clearer and shorter, stop operating on the individual command level, and start thinking in a procedural mode, using repeats, sub-routines, or procedures. This task was administered in order to examine how the different children would approach a program that already worked well, and produce a different program for the same purpose by using higher-level units. In other words, when asked to read this program again, the child had to re-scan the code and mentally simulate what the program was doing overall, rather than line by line, see how the goals of the original program were achieved, and how the program could be written in another way. This is a very important aspect in analyzing children's knowledge of programming. Pea and Kurland (1987), for example, believe that just as we cannot expect students who can barely decode or comprehend text to be proficient writers (p.167), we also cannot expect students with a low-level ability in reading, decoding, and understanding programs to be able to write functional higher-level programs or gain insights into other domains.

On the whole, the experimental children were more flexible than the others and attempted to explore several different ways for producing the same drawing. They reached a more modular level of code, and most of them used repeats, sub-procedures and variables.

The experimental children also performed significantly better than the control children on the three other items of this test, covering use of inputs, modification of procedures according to specific requests, and prediction of results of short but confusing graphics programs.

Examples from the Logo Computer Test: This test included five related sub-tasks:

Task:	Skills Assessed:
T_1	Debugging: Run the program, find the bugs, fix the code on the computer
T_2	Debugging: Fix the same code on paper
T_3	Modularization of the given code
T_4	Optimization of the code according to specific requests
T_5	Higher level of Optimization

In tasks 1 and 2, the children were asked to solve problems related to debugging. On the whole, the experimental children showed that they developed debugging skills since they performed much better on these two sub-tasks, were faster at identifying the bugs, were able to locate the bugs in the given program on the computer and on paper, and then succeeded in re-evaluating the program, and creating an output that corresponded perfectly with the original goal given to them in that task. The following data table shows the results for tasks 1 and 2, which required that the children run the given program, analyze the features of the resultant graphics, dentify the discrepancies between them and the desired graphics, enter the Logo code on the computer, locate the different bugs causing the discrepancies, fix the program on the computer, and finally, add the corrections on the program that was written on the paper. Results from the debugging tasks for all the 51 children:

	no. of bugs found and fixed	identify & fix bugs in computer prog.	identify & fix bugs in paper program	Average time for solving 1&2
Experimen. class N=17	16 - all bugs 1 - one bug	17 children - yes 100% succeeded	17 children - yes 100% succeeded	15 minutes per child
Control 1 N=18	9 - all bugs 4 - one bug 5 - none	13 children - yes 5 children - no 70% succeeded	8 children - yes 10 children - no 44% succeeded	35 minutes per child
Control 2 N=16	2 - all bugs 4 - one bug 10 - none	6 children - yes 10 children - no 37% succeeded	2 children - yes 14 children - no 12% succeeded	55 minutes per child

This data table speaks for itself. The superiority of the experimental children (on all math levels) over the other pupils is clear, as is that of C_1 over C_2.

In debugging, the first thing all the experimental children did was to change the HT (Hide Turtle) command at the very beginning of the procedure, to ST (Show Turtle), so that they could follow the turtle as it executed the code. On the other hand, the first strategy that most of the children in C_2 and many in C_1 used was to copy the program given to them on paper (in sub-task 2) into the Logo Command Center and execute it line by line. This strategy worked well until they reached the REPEAT statements, which were written on more than one line. Then, the children got confused because the program did not work, and they were sure that they had located a bug; they "fixed the bug" but, of course, with little success. Instead of trying a new strategy, these children then erased everything and started to copy the procedure into the computer in "direct mode" again, which resulted in the same thing happening again, and so on.

In tasks 3 and 4, the children were asked to optimize the code given to them in tasks 1 and 2, and make it clearer and shorter. In order to solve these sub-tasks, the children had to cease operating on the individual command level, and start thinking in a procedural mode, using REPEATs, procedures, and later inputs. As in question 4 of the Logo Pencil-&-Paper Test, the children were asked to optimize code they had just debugged. One hundred percent of the experimental children produced adequate modifications; but only 66% of C_1 and 25% of C_2 were able to do so, with 6 children in C_1 and 12 children in C_2 failing completely.

In task 5 the children were asked: "Can you make a program that draws the same picture, but each flag will be in a different color?" Ten experimental children, and two children from C_1 managed to solve this task. Five experimental children used a variable even for the color (instead of using RANDOM).

To summarize, the experimental children were more flexible and attempted to explore a greater variety of ways for producing the same drawing. They understood and reached a more modular level of code, and many of them tried to use repeats, sub-procedures, and variables. Interestingly, all the experimental children, who had already performed much better than the control children in the similar task (question 4 of the Logo Pencil-&-Paper Test), performed even better here using the computer. But the children from C_2 got more confused at the computer, and performed less well than they had on the Pencil-&-Paper task. C_1 was somewhere in between: the high-math children, like those from the experimental class, performed much better at the computer, and the medium- and low-math children performed similarly to those from C_2, that is, far less successfully than they had in the Pencil-&-Paper task.

Results from the FRACTIONS POST-TESTS

All the teaching of fractions, for all the three classes, was conducted for two months, during regular math lessons only and following the city-wide curriculum and traditional teaching methods. The experimental class was not provided with any additional formal instruction on fractions, although several focus sessions treated representations of fractions in the context of discussing instructional design.

Results from the Rational-Number Pencil-&-Paper Test: The post-test included 65 questions. Out of these, 60 were taken from the Rational-Number Project (RN Project, Lesh et al., 1983 pp. 309-336). The remaining five were designed by the researcher and included word problems and construction of representations. Of the 60 RN Project questions, 43 were given to the children in the pre-test, then again in the post-test.

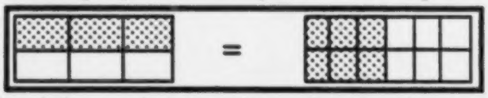
As an example, the following diagram shows the average percentages of correct answers in the pre-tests and the post-tests of each math group, and within each class. The difference in pre- and post-test scores of the children from the experimental group was almost twice as great as in that of the children from C_1, and two-and-a-half times as great as in that of C_2.

RATIONAL NUMBER PENCIL AND PAPER TEST

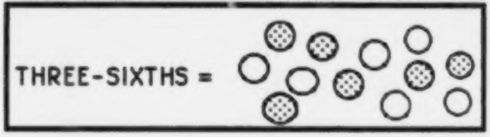
% Correct on the SAME 43 pre- and post questions	HIGH		MEDIUM		LOW	
	Pre	Post	Pre	Post	Pre	Post
EXPERIMENTAL	57%	82%	51%	71%	41%	59%
CONTROL 1	64%	78%	31%	62%	40%	55%
CONTROL 2	53%	58%	46%	59%	37%	50%

Lesh et al. (1983) tested their subjects on different "translation modes" between representations of fractions. In their large-scale testing program, they found that some translations were more difficult to process than others. They are grouped below, from easiest to hardest.

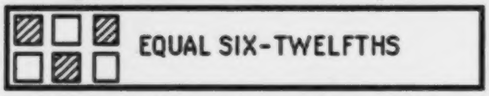
1. Translating word representation into word representation (three-sixths equal six-twelfths).
2. Translating symbols into symbols (for example, 3/6 = 6/12).
3. Translating symbols into words (for example, 3/6 equal six-twelfths).
4. Translating words into symbols (for example, three-sixths = 6/12).
5. Translating a picture into a picture, for example,



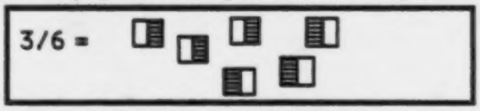
6. Translating words into a picture, for example,



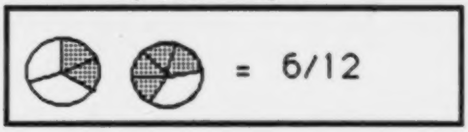
7. Translating pictures into words, for example,



8. Translating symbols into pictures, for example,



9. Translating pictures into symbols (the hardest for children of all ages), for example,

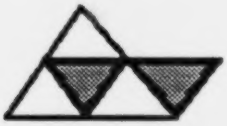


The scores of the children in the RN Project were taken as standard scores (in their percentage of correct answers) for the present study. Thus, the children from the experimental group (N=17) were compared not only with those from the two control classes (N=18, N=16), but also with the fourth-grade children from the RN Project (N= 43).

We gave specific attention to the analyses of the most difficult translation modes between rational-number representations, that is, levels 6 through 9 above. These four levels of translations were the most difficult for children of all ages in previous strudies, and were equally so for all children in the present study's pre-tests. In the post-tests however, these four translation modes were still relatively difficult for the control children, but the experimental children dramatically improved.

Let us consider an example. Lesh et al. considered question 50 to be so complex that it was not given at all to the fourth graders in the RN Project, only to sixth, seventh, and eighth-grade children (Lesh et al., p. 326). The children were asked to translate a pictorial representation into a written (verbal) representation of a fraction:

50) WHAT IS THE DENOMINATOR OF THE FRACTION THAT TELLS US WHAT PART OF THE PICTURE BELOW IS SHADED?



a. five-thirds b. five c. three
d. two e. not given

This question presented children with a polygonal region representation, with a numerator that was higher than 1, a denominator, a representation of a rational number lower than 1, in a discrete object that included a perceptual distraction (i.e., one part was "outside" the triangle area). In order to choose one of the options, the children had to 1) translate the given picture into symbols or words (two fifths are shaded in), 2) read the question again and realize that the question referred to the denominator of the shaded fraction, and 3) find the correct answer, which was b. Option a. is confusing because it is written like a spoken symbol and includes "relevant" numbers--five and thirds. Option b. is confusing because it do not mention "fifths," but rather "five" (the denominator is "five").The following diagram shows the scores in their percentage of correct answers for question 50.

	EXPER.	CTR 1	CTR 2	STAND
HI	88%	78%	16%	Not given to 4th. 7th gr. got
MD	60%	16%	25%	
LO	50%	0%	40%	
AVE	66%	31%	27%	26%

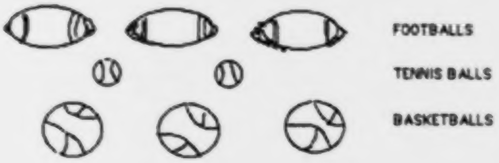
% CORRECT ON QUESTION * 50 TRANSLATING THE DENOMINATOR IN A PICTURE SHOWING TWO FIFTHS, INTO THE WORD "FIVE."

The experimental children's scores were twice as good as those of the control children, and even twice as good as those of the seventh graders from the RN Project.The low- and medium-levels of the experimental class did much better in this question than the low- and medium-level of C_1 and all the levels of C_2.

Perhaps there is some "transfer" from Logo programming experience at work here. Decomposing a given picture into its geometrical components is a common process in Logo programming, and a skill children usually acquire in their ongoing programming experiences. What Lesh et al. (1983) and Behr et all. (1983) consider as a "perceptual distraction" (i.e., the one little triangle that was "outside" the big triangle area) was probably not at all a distraction for the children who looked at the picture with "Logo eyes" and decomposed it into its five geometrical components.

Question 42, involving translation of pictorial into symbolic representation (eighth level of difficulty above) is another example.

42) WHAT FRACTION OF THE BALLS ARE TENNIS BALLS?



a. 2/8 b. 3/2 c. 2/6
d. 6/2 e. not given

This question, number 42, was the thirteenth most difficult of the 18 asked in this subset. It was the forty-fourth most difficult in the whole set of 60 questions given in the RN Project to children from fourth through eighth grades (Lesh et al., 1983, p.323). It included a discrete object representation in which the represented rational number was less than one; moreover, parts of this object were not congruent and were visually distracting. The diagram shows the scores (given as percentage of correct answers) on this question:

	EXPER.	CTR 1	CTR 2	STAND
HI	100%	72%	72%	
MD	72%	68%	50%	
LO	50%	40%	20%	
AVE	74%	60%	47%	35%

% CORRECT FOR QUESTION # 42. TRANSLATING A RATIO REPRESENTED IN A PICTURE, INTO ITS SYMBOLIC REPRESENTATION.

As seen above, none of the high-math experimental children made any mistakes. The medium-math experimental children scored like the high-math children in the two control classes. The experimental class as a whole did twice as well on this subset than the children in the RN Project, and 14% better and 27% better than C_1 and C_2 respectively.

On the whole, the trends that were found in the results of the Logo post-tests were also found in this post-test: the experimental children consistently scored higher than the other two classes; and C_1 usually scored higher than C_2. The high-math children from C_1 make up a special group. They were never as good as the high-math experimental children, but most of the time they were as good as the experimental medium-math children. Their scores are often higher than those of the children from the RN Project, and stand out from those of the other control children.

What does this mean? It seems as though only the high-math children in C_1 strongly benefitted from Project Headlight experience with respect to the pictorial-to-symbolic translation. This is probably due to their programming expertise, which contributed to their ability to translate picture representations into written ones, and vice versa. This phenomenon requires further investigation. It is an interesting one, since it suggests a correlation between the children's level of understanding and involvement in Logo programming, and their ability to understand and use different representational systems.

Results from the Boston Curriculum Referenced Test, Math Level 4: On June 15 and 16, all the pupils were tested in math, as part of their end-of-year public school series of referenced tests. This mathematics test included 40 multiple-choice questions. The average number of incorrect answers per child on the whole test (i.e., for all 40 questions) was 5.06 incorrect answers per child in the experimental class, an average of 6.27 incorrect answers per child in C_1, and an average of 9.45 incorrect answers per child in C_2.

Of the 40 questions, six were specifically on fractions ordering and equivalence, four on decimals; four on measurement of distance and time that included fractions, and one on understanding geometrical shapes (i.e., this was the subset of 15 questions directly related to rational-number concepts, their representations and computation). The average of incorrect answers per child on this subset of 15 rational-number questions was only 1.60 in the experimental class, an average of 3.16 incorrect answers per child in C_1, and an average of 4.62 incorrect answers per child in C_2.

Several conclusions can be drawn from analyzing these results. The first is that the experimental children, in general, did much better on the entire conventional school test than the two control classes.

The second conclusion is related to the children's incorrect answers in the rational-number concepts subset of this test. In the experimental class, only 29% of the incorrect answers in the whole test (40 questions) were incorrect answers about rational-number concepts . But in both C_1 and C_2, approximately 50% of the incorrect answers were on rational-number concepts (i.e., in C_1 the average proportion was 3.16 to 6.27; in C_2 it was 4.62 to 9.45). This again shows the clear superiority of the experimental class on rational-number knowledge as measured by this standard test.

The third conclusion is related to transfer. By subtracting the average of incorrect answers on the fractions subset from the average of incorrect answers on the whole test, we can examine the children's average of incorrect answers to all the non-fractions questions: for the experimental class, 5.06 - 1.60 = an average of 3.46 incorrect answers per child on non-fractions questions; for C_1, 6.27 - 3.16 = 3.11; and for C_2, 9.45 - 4.62 = 4.83. The differences between the experimental class and C_1 are not significant here, but that the differences between these two classes and C_2 are. This finding is interesting because it might be that the experience of Project Headlight children (experimental class and C_1) with Logo programming contributed to their general mathematical ability. The results from the school's math test are shown in the following diagram:

class:	Average of incorrect ans. per child on all 40 quests.	Average of incorrect ans per child on the subset of 15 rational-number quest.	proportion of inc. ans. in subset/whole
Experimental class	an average of 5.06 incorrect ans. per child	an average of 1.60 incorrect ans. per child	1.60 to 5.06 = 29%
Control class 1	an average of 6.27 incorrect ans. per child	an average of 3.16 incorrect ans. per child	3.16 to 6.27 = 51%
Control class 2	an average of 9.45 incorrect ans. per child	an average of 4.62 incorrect ans. per child	4.62 to 9.45 = 48%

Results from the Public Schools Curriculum Referenced Test, Math, Level-4

Discussion and Conclusions

Logo

The children's intensive use of the Logo programming language to design instructional software on fractions, over the four-month period, was an effective way to learn Logo programming concepts and skills. The ISD learning approach that was implemented in this study was strongly inspired by and very similar in nature to Papert's approach to programming (e.g., 1980, 1986), and different from the typical "Logo course" or "Logo curriculum" approaches, the predominant approaches used in schools today or in Logo research over the last decade (e.g., Pea &

Sheingold, 1987; Soloway, 1984; Carver, 1987; Perkins et al., 1985; or Heller, 1986). ISD took Papert's ideas one step further. It consisted in children's using Logo for the purpose of designing and programming instructional software. The length and continuity of ISD (compared to several days to three weeks for typical Headlight projects) together with its structure and complexity enhanced the experimental children's learning of Logo. In addition, the children in ISD had their minds on other issues besides "a program that works." They worked on their project while thinking about teaching other pupils, about screen designs, about interactivity and feedback--much like professional software designers.

The experimental children's actual pieces of software (Harel, 1988a) demonstrated that they were able to generate a great amount of code in an open-ended way during their software production, but we also saw how they were able to do so very efficiently and precisely, following the pre-defined constraints and requirements of the Logo post-tests, which showed that the experimental children successfully acquired debugging skills. In their discussion of why children usually fail to acquire these skills in the course of learning elementary programming, Carver and Klahr (1986) give three main reasons: "a) Debugging is a complex skill; b) it requires extra capacity; and c) it is rarely taught directly" (pp. 27-30). In light of ISD results, the third, should be revised. Children who study programming under rich learning conditions should be able to acquire basic debugging skills by frequently exploring and experiencing debugging situations as they learn to program.

Experimental children did not receive anything like Carver's (1987) instructional method on learning debugging skills, but went through many phases of designing and debugging in the course of making instructional software. There were many situations when children designed something in their Designer's Notebooks in one way, but when they actually programmed it found discrepancies between the original drawing and the program's actual output. In some cases children serendipitously took advantage of the unexpected outcome: They re-evaluated the original plan or design and decided to go on with whatever had appeared on the screen (in other words, children modified their plan instead of treating the output as a result of discrepancies or bugs). However, in other cases, children felt a strong need to locate the bugs and fix them so that the program's output would match the original design drawn in their notebook. Through this combination of thinking and programming strategies, the experimental children learned to become good debuggers, as well as creative and flexible problem-solvers.

To summarize, the experimental children in this study did not find it difficult to learn Logo--in fact, they learned a great deal of Logo and enjoyed it tremendously. They were not offered explicit and direct instruction, and they did not complete well-defined exercises or worksheets that highlighted or simplified complex Logo concepts; rather, banal as it may sound, they learned Logo by working intensively and productively with it. Their programming activity was integrated into a larger context of software design and development, opportunities for extensive practice combined with meaningful objectives helped experimental children to learn and understand Logo at a higher level than other Headlight children, and at a much higher level than C_2 children. In short, the children's involvement in a rich, meaningful, and complex task, working towards designing and programming a "real" product for "real" people, enhanced their understanding of what Logo is and their knowledge of how to use it.

The great differences between the experimental class and C_2 had been predicted from the beginning. The children from C_2 learned programming once a week in the school's computer lab and were not involved in large-scale, meaningful, or complex programming projects. This type of instruction, we should remember, is typical of most published Logo research. Many "Logo-programming courses" have been developed, and many Logo exercises and worksheets created, in view of an isolated "Logo curriculum" to be used in a class like C_2. But many of these, in fact, miss the purposes and the major ideas of Logo (or programming in general), reduce the creative learning possibilities offered to young children though Logo programming, and minimize the use of Logo as a thinking and learning or "knowledge reformulation tool."

Although the differences between C_2 and the experimental class had been expected from the start, this was not the case with C_1 and the experimental class. Since both of these classes were part of Project Headlight, no one could really know whether they would show any difference in their knowledge and understanding of programming by the end of the experiment. However, there were many differences, which proved to be quite interesting. It seems that the experimental conditions helped develop a higher-level knowledge and expertise in Logo programming for all the experimental children on all math levels. On the other hand, Project-Headlight conditions also developed some kinds of expertise among C_1 children, since after all they consistently scored higher than those of C_2. But when compared to the experimental class, only the high-math children of C_1--and then only occasionally--achieved the average level of Logo understanding of the experimental children; whereas the medium and low-math children from C_1 did not seem to benefit as much from Project Headlight conditions. The results from the Logo post-tests therefore demonstrate that the experimental children, working under the ISD conditions, were those who improved most dramatically in their Logo programming knowledge and skills.

Fractions

The design and programming of representations for fractions, and the creating of graphical displays for equivalence and operations of rational numbers, influenced children's thinking and learning about representations of fractions and the relationships between them. The children's goal of representing and explaining fractions to another person helped them to overcome their own difficulties in understanding fractional representations (which had been identified in the pre-tests and reported in other studies about fractions; e.g., Behr et al. 1983; Lesh et al. 1983; or Tierney, 1988).

In traditional approaches to learning fractions, emphasis in placed on learning algorithms (for finding the greatest common denominator and for arithmetic operations). But algorithms are only the surface structure of rational-number knowledge. Interestingly, ISD's emphasis on understanding the deep structure of representations transferred to significant gains in the experimental children's handling of algorithms on the conventional test. Thus, indirect learning (emphasizing acquisition of deep structure) proved more successful than traditionally-styled direct learning.

Unlike previous projects, the present study did not attempt to identify the "best" manipulative aids for the fractions learning process. It emphasized, however, that while working with Logo, children could decide by themselves which representations they wanted to create and manipulate. Since many representation for fractions could be programmed in Logo, children could choose which ones they felt comfortable with, and which ones they wanted to learn, work with, or combine. The computerized Logo environment proved to be rich in its potentially available representations (i.e., the representations were not pre-programmed, but the children wrote their own procedures for many representations) and in making it possible for children to combine several representations at the same timeon the same screen.

In the end, hardly anyone in the experimental group programmed more than fifteen representations. The experimental children did not use any manipulative aids, nor did they cover a very wide range of rational-number sub-constructs, or translate or manipulate a large number of representations. However, the post-tests did include a large number of rational-number subconstructs, their multiple representations (in the pencil-and-paper test), and a variety of manipulative aids (in the interview). The experimental children did well on these tests, indicating successful indirect learning from producing software about fractions. It was not because they had actually learned with these materials that they succeeded, but because--through designing and programming instructional software about fractions--they thought about and experienced the principles involved in the relationships between the different rational-number concepts and their representations.

Suggestions for Future Research and Development

ISD offers changes in the approach to the learning and teaching of Logo and fractions concepts and skills, and of several important and more general thinking and problem-solving skills. But it also proposes a more general approach that could be implemented with other topics in mathematics (e.g., proportional reasoning), and in the elementary-school curriculum in general. Children should be able to learn areas of science, geography, literature, art, and social studies through this method of instructional software design. Further experiments should be conducted for investigating which topics lend themselves best to this approach, and how this method contributes and changes children's understanding of these topics.

A mixture of research methods was used to study children's open-ended and extended problem-solving processes. This marked a shift away from current methodology in the field, which concentrates on analyzing shorter and more narrowly-defined problem-solving sessions. As a result, we were able to collect data that were informative at many levels. All the experimental children succeeded at the same "job," producing a piece of instructional software, each in his or her own way. As shown in previous findings on Logo (e.g., Turkle, 1984; Weir, 1987; Lawler, 1985; Papert, 1979, 1980), software design also proved to be a learning activity that allowed for individual differences in learning, mastery, and expression. New research methods are needed for identifying, analyzing and describing individual differences in styles of expression and mastery, which could for example, trace differences in individual learning processes and production strategies and relae them to different technological tools and learning theories.

ISD demonstrated that children--given a context of adequate structure and control over their learning--can function much like a community of adult professionals. They shared ideas, criticized each other's products, and often provided help to their peers; yet, in the end, they each produced their own product. An interface system similar to the one proposed here would encourage children along this path of greater responsibility for their individual work and enhanced collaboration with others, supporting individual and social constructionism. However, no "technological fix" can bring about the evolution of a learning community. Children in the ISD project, and in Project Headlight generally, 9 had free access to computers they could appropriate and use as tools in various and flexible ways and for exploring ideas of their own as well as different school subject areas. Many factors, including the complexity and open-endedness of the project, the children's independent work styles, the project's length, and the enthusiastic and constructive working relationship among researcher, teacher, and students combined to create an essentially new approach to learning. Dewey, Papert, and others have taught us to question our assumptions about children's limitations. ISD freed children from constrained images of themselves as learners and fostered intellectual excitement, creativity, and a sense of commitment to their work. The small learning community that emerged over the four months of the ISD project exemplified these qualities and simultaneously produced success, even as measured in conventional ways.

Footnotes

1. The Designer's Notebook contained a minimal structure to aid children in their work, reflections, and planning skills. The rationale behind teaching children to plan, reflect, and take notes can be found several theories and reports of experiments in educational psychology literature, more specifically, in the literature on learning metacognitive, cognitive control, and related thinking skills (e.g., Chipman et al., vol. 1 & 2, 1985; and Brown, A.L. et al., 1983). In these experiments, however, children were not required to plan and reflect in writing every day for as long a period as they were in ISD.
2. This requirement helped the project to fit into the class and school schedule and made possible estimates and generalizations about what children could accomplish in similar projects in the future.
3. For example, in two of these discussions, we hung two posters, one on each side of the blackboard. On one poster we wrote, "WHAT IS DIFFICULT ABOUT FRACTIONS?" and on the other, "WHAT SCREENS OR REPRESENTATIONS COULD BE DESIGNED TO EXPLAIN THESE DIFFICULT CONCEPTS?" We asked the children to generate ideas for both posters simultaneously. We wrote down their suggestions and gave them time to take notes on them in their Designer's Notebooks. If they wished, of course, they could use these ideas as inspiration for generating new ones.
4. For example, the teacher, the researcher, or one of the children, would stand next to one of the classroom computers, in front of the whole or part of the class and explain how to use REPEAT, IFELSE, variables, etc. The children could take notes on concepts and programming routines in their Designer's Notebooks, or proceed directly to their computers to write a procedure including the new programming skill or concept.
5. There has been a good deal of discussion, inside and outside the "Logo community," of how intrinsic these experiences are to Logo itself, or to possible Logo cultures. The latter cultural view is now dominant.
6. The reasons for their selection, and a detailed description of the characteristics of the pupils, their teachers, and their different fractions and Logo learning approaches can be found in Harel (1988a).
7. Project Headlight is directed by Seymour Papert, Professor of Media Technology, at Massachusetts Institute of Technology. See also Papert (1985, 1986, 1987).
8. The results from each of these questions are described in detail in Harel (1988a).
9. For example, Lego, LOGO, and Design, by Resnik, Ocko, and Papert, 1988.

Bibliography

Behr, M. J., Lesh, R., Post, T. R., & Silver, E. A. (1983). Rational Number Concepts. In Lesh, R., & Landau, M. (Eds.), *Acquisition of Mathematics Concepts and Processes*. New York: Academic Press.

Behr, M. J., Wachsmuth, I., Post, T. R., & Lesh, R. (1984). Order and Equivalence: A Clinical Teaching Experiment. *Journal of Research in Mathematics Education, 15* (5), 323-341.

Bennett, A. B. (1981). *Fraction Bars: A Step By Step Teacher's Guide*. University of New Hampshire & Scott Resources Publishers.

Brown, A. L., Bransford, J. D., Ferrara, R. A., & Campione, J. C. (1983). Learning, Remembering, and Understanding. In W. Kessen (Ed.) *Handbook of Child Psychology: Cognitive Development, Vol. 3*. New York: Wiley.

Carpenter, T. P., Coburn, T. G., Reys, R. E., & Wilson, J. W. (1976). Notes from the National Assessment: Addition and Multiplication with Fractions. *Arithmetic Teacher, 18(4)*. 245-249.

Carver, S. M. (1986, December). *Transfer of LOGO Debugging Skill: Analysis, Instruction, and Assessment*. Ph.D. Thesis. Pittsburgh, PA: Carnegie-Mellon University, Department of Psychology.

Carver, S. M., & Klahr, D. (1986, February). Assessing Children's LOGO Debugging Skills With a Formal Model. Pittsburgh, PA: Carnegie-Mellon University, Department of Psychology. (In Press: *The Journal of Educational Computing Research*).

Chipman, S. F., Segal, J. W., & Glaser, R. (1985). *Thinking and Learning Skills. Vol. 1 & 2*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Clements, D. H. (1985, April). *Effects of Logo Programming on Cognition, Metacognitive Skills, and Achievement*. Paper presented at the American Educational Research Association, Chicago, Illinois. (In Press).

Clements, D. H., & Gullo, D. F. (1984). Effects of Computer Programming on Young Children's Cognition. *Journal of Educational Psychology, 76(6)*. 1051-1058.

Davidson, P. S.(1977). *Idea Book for Cuisenaire Rods at the Primary Level*. New Rochelle NY: Cuisenaire Company of America.

Davidson, J. (1969). *Using The Cuisenaire Rods--A Photo-Text Guide for Teachers*. New Rochelle NY: Cuisenaire Company of America.

Dewey, J. (1902). *The Child and the Curriculum*. In R. D. Archambault (Ed.), John Dewey on Education. Chicago: University of Chicago Press (1974).

ETC (1985, November). Technical Report no. 85-21, written by Davidson, P. S., Dickenson, A. M., & Tierney, C. C. *"Pies are hard to find out about..." An Inquiry into children's understanding and nature of fractions*. Harvard Graduate School of Education, Cambridge: The Educational Technology Center.

Feurzeig, W., Horwitz, P., & Nickerson, R. S. (1981, October).*Microcomputers in Education*. Report no. 4798. Cambridge Ma: Bolt Beranek & Newman.

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1969). *Programming Languages as a Conceptual Framework for Teaching Mathematics*. Technical Report no. 1899. Cambridge MA: Massachusetts Institute of Technology and Bolt, Beranek, & Newman.

Harel, I. (1988a, June). *Software Design For Learning: Children's Construction of Meaning for Fractions* *and Logo Programming*. Ph.D Thesis. Cambridge, MA: Media Laboratory, Massachusetts Institute of Technology.

Harel, I. (1988b, in progress). *Debbie as a Software Designer*. Cambridge, MA: Media Laboratory, Massachusetts Institute of Technology.

Harel, I. (1988c, in progress). *Patterns in Instructional Software Design: Case Studies of Young Children*. Cambridge, MA: Media Laboratory, Massachusetts Institute of Technology.

Harel, I. (1986, July). *Children As Software Designers: An Exploratory Study in Project Headlight*. Paper presented at the LOGO 86 International Conference, Cambridge, MA: MIT.

Harrison B. (1972). *Pattern Blocks Activities for Children*. Weston, MA.

Heller, R. S. (1986). *Different Logo Teaching Styles: Do They Really Matter?* Paper presented at the First Workshop of Empirical Studies of Programmers. Washington, DC.

Janvier, C. (Ed.) (1987). *Problems in Representation in the Teaching and Learning of Mathematics*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kurland, D. M., & Pea, R. D. (1983). Children's Mental Models of Recursive Logo Programs. In *Proceedings of the Fifth Annual Cognitive Science Society*. Rochester, NY: Cognitive Science Society.

Lammer, S. (1987). *Programmers At Work*. Redmond, Washington: Microsoft Corporation Press.

Lesh, R.,& Landau, M. (1983). *Acquisition of Mathematics Concepts and Processes*. Academic Press.

Lawler, R. W. (1985). *Computer Experience and Cognitive Development: A Child Learning in a Computer Culture*. West Sussex, England: Ellis Horwood Limited.

Papert, S. (1987). *Using Computers to Combat Illiteracy: Towards a Constructionist Theory of Creative Learning*. A Proposal to the MacArthur Foundation. Cambridge, MA: the Media Tech. Lab.

Papert, S. (1986). *Constructionism: A New Opportunity For Elementary Science Education*. A Proposal to the National Science Foundation. Cambridge, MA: the Media Technology Laboratory, MIT.

Papert, S. (1985, July). *Computer Criticism vs. Technocratic Thinking*. Cambridge, MA: the Media Technology Laboratory, MIT. Also appeared in LOGO 85 Theoretical Papers. 53-67.

Papert, S. (1984a). *Microworlds Transforming Education*. Paper presented at the ITT Key Issues Conference, Annenberg School of Communications, University of Southern California.

Papert, S. (1984b). *New Theories for New Learnings*. Paper presented at the National Association for School Psychologists' Conference.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). *Final Report of the Brookline Logo Project. Parts I and II*. LOGO MEMO no 53. Cambridge, MA: MIT.

Pea, R. D. (1984). *Symbol Systems and Thinking Skills: Logo in Context*. Paper presented at the LOGO 84 Conference, Cambridge MA: MIT. Appeared in the LOGO 84 Pre-Proceedings Book. 55-61.

Pea, R. D., & Kurland, D. M.. (1984). On the Cognitive Effects of Learning Computer Programming. *New Ideas In Psychology, 2(2)*. 137-168.

Pea, R. D., and Sheingold, K. (Eds.) (1987). *Mirrors of Mind: Patterns of Experience in Educational Computing*. Norwood, NJ: Ablex Publishing Corporation.

Peck, D. M., & Jencks, S. M. (1981). Conceptual Issues in the Teaching and Learning of Fractions. *Journal for Research in Mathematics Education. 12(5)*. 339-348.

Perkins, D. N. (1986). *Knowledge As Design*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Perkins, D. N. & Martin, F. (1985, October). *Fragile Knowledge and Neglected Strategies in Novice Programmers*. Technical Report no. 85-22. Cambridge MA: Educational Technology Center, Harvard Graduate School of Education.

Post, T. R., Wachsmuth, I., Lesh, R., & Behr, M. J. (1985). Order and Equivalence of Rational Numbers: A Cognitive Analysis. *Journal on Research in Mathematics Education, 16(1)*. 18-36.

Post, T. R. (1981). Results and Implications from the National Assessment. *Arithmetic Teacher, 28(9)*. 26-31.

Resnik, M., Ocko, S., and Papert, S. (1988). Lego, LOGO, and Design. To appear in *Children's Environments Quarterly*. Spring 1988.

Salomon, G., & Perkins, D. N. (1986, February). *Transfer of Cognitive Skills from Programming: When and How?* Technical Report no. 2. Tel Aviv, Israel: Unit for Communication and Computer Research in Education. Tel Aviv University.

Solomon, C. (1986). *Computer Environments for Children: A Reflection on Theories of Learning and Education*. Cambridge, MA: MIT Press.

Tierney, C. C. (1988). *Construction of Fraction Knowledge: Two Case Studies*. Unpublished Doctoral Thesis. Cambridge, MA: Harvard Graduate School of Education.

Turkle, S. (1984). *The Second Self: Computers and the Human Spirit*. New York: Simon and Schuster.

Watt, D. (1979). *Final Report of the Brookline Logo Project. Part III: Profiles of Student's Work*. Logo memo no. 54, A.I. Memo no. 546. Cambridge MA: Artificial Intelligence LAB, MIT.

Weir, S. (1986). *Cultivating Minds: A Logo Case Book*. New York: Harper & Row.

The research reported here was conducted at Project Headlight's Model School of the Future during 1987-88 as part of my Ph.D. Thesis at the MIT's Media Laboratory; and was supported by the IBM Corporation (Grant # OSP95952), the National Science Foundation (Grant # 851031-0195), the McArthur Foundation (Grant # 874304), the LEGO Systems A/S, and the Apple Computer Inc. The preparation of this paper was supported by the National Science Foundation (Grant # MDR 8751190). The ideas expressed here do not necessarily reflect the positions of the supporting agencies. Much thanks to the teachers and students of Headlight, without whom this research would have not been possible.

LINGUISTICS, SCIENCE, AND MATHEMAITICS FOR PRE-COLLEGE STUDENTS: A COMPUTATIONAL MODELING APPROACH

E. Paul Goldenberg
Education Development Center, Inc.
55 Chapel Street
Newton, MA 02160

Science and language are not traditionally partners in curricular taxonomies from elementary school on up through the universities. But, with the computer as a modeling tool, students can engage in a kind of sophisticated exploration of language that has important implications for both science and language study. The modeling, itself, is a fundamentally mathematical activity. This paper explores the contribution that language study can make to childrens' pre-college scientific development.

Science, science's image, and science education

Using the word *behaviors* just a bit loosely, science is the attempt to discover, describe, and predict the behaviors of naturally occurring things. In casual use, especially in schools, *science* evokes images of rocks and stars, plants and animals, levers and pulleys, mixing chemicals. Scientists study the social rules that govern chimpanzees in the wild, or seek, through theory or experiment, the fundamental building blocks of matter.

Most of this territory is absolutely inaccessible to children first hand. Kids can't get particle accelerators or wild chimpanzees in school and wouldn't have the time, money, and mathematical sophistication to make much use of them anyway. Even if a child thinks he notices a pattern in the data that he has been memorizing about the solar system—this particular incident occurred in a classroom in which I was once substituting for a colleague—the child cannot check out his idea unless the book he happens to have contains the data he happens to need. So, for the most part, school science becomes the acquisition of facts that have been found by people who call themselves scientists. There are 92 elements (wrong); protons are indivisible (wrong again); one planet has rings (oops, more), and so on....

Things are not, of course, as bleak as that in all science curricula. Some 20 years ago the Elementary Science Study (ESS) began producing a series of highly exploratory activities through which children could engage in real scientific exploration, discovery, theorizing, and verification, using safe, simple, inexpensive, and available materials. These curricula are still in print, and still exciting, but still, alas, relatively little used.

Probably the single greatest reason that ESS-like curricula are not more widespread—certainly a reason that is often expressed—is that the management of experimentation with diverse manipulatives is more effort than many schools or teachers believe it is worth. But the problem here is not laziness. Teachers, as a group, often work above and beyond their contractually defined obligations when they believe their effort will benefit their students. In fact, teachers *do* have students manipulate materials, and *do* face the management issues, but the materials are more likely to be used for demonstration of learned facts—papier maché models, dioramas, and the like— than for experimentation. So if the problem is not an unwillingness to expend effort managing the manipulation of messy materials, then what is it? The popular image of science and scientists suggests an answer: the problem is a fundamental belief that the activities of scientists are impossible for non-scientists, and that the acquisition of scientific "facts" is all that is left for us. The consequence is a disbelief that experimentation really adds much. The image of science may be so poisoned by its perceived inaccessibility—stars and atoms are just too far from our personal knowledge and daily experience—that anything in the traditional domain, even if its materials are available, feels cold, removed from personal meaning, and most importantly, either unknowable or already known. Why bother with materials, after all, if what one *perceives* as the purpose of science—learning facts—can be fulfilled just as easily from a book?

Language as a domain in which students can be scientists

How, then, do we infuse the sense of scientific inquiry into science education and rekindle a sense of personal meaning?

One answer begins by looking at children's natural inclinations. Children are already, in one sense, very good scientists. For the youngest children, science and play are not so different. The language play of pre-schoolers provides a good example. Some of that play—including the often amusing and sometimes irritating repetitions of *our* language—constitutes a kind of experiment. Except for the lack of *articulation* of the rules that have been discovered (and the related lack of formalized plan of the experiments), this is science at its best. Children naturally engage themselves in figuring out how things work and apply themselves just as diligently to figuring out the social rules of a group as to learning what all the controls on the car or stereo do. We even explicitly acknowledge the social-scienties in our children when we characterize their infringements of our social rules not as perversity but as "testing" the limits, experimenting to see what we will do.

The task of science education, then, is, in part, to excite children about formalizing what is for them already a familiar process—to help them articulate what they know and, especially, how they go about knowing.

To make such a formalization feel natural and enjoyable, we must look for subject matter that children can investigate without fancy equipment, advanced math, unnatural abstraction, or vast sums of money and time.

Language is just such a subject. As an object of scientific inquiry, it has two additional advantages. It is already part of the curriculum, and therefore does not further burden an already overcrowded schedule. Also, it is about as personal and accessible an object of inquiry as one can find. Each of us has, in his or her own head, an unimaginably vast store of data about our language, and we have fairly ready access to these data. We also have theories to test. Before we ever set foot in school, we all began developing theories to organize the otherwise unmanageable heap of data. For example, the 18-month old is likely to utter words like *caught* and *feet* but abandon them later for words like *catched* and *foots* because these new versions (which the child has never heard an adult use) accord better with her newly-developed theory of how language works. Only after sufficiently many "exceptions" to her theory turn up will the exceptions warrant a paradigm shift or a new theory of their own. Then the child reverts back to the correct forms that she had once before used.

In our day-to-day *use* of language, we remain unaware of our theories. But adults and older children who have the cognitive ability to organize, formalize, and articulate such rules, can be helped to recognize and explore them. For them, language becomes a perfect topic for scientific investigation.

That is where the computer comes in. A high-school senior with a distinctly (but, alas, not uncommonly) low impression of her own math/science knowledge and aptitude was exploring language in a computer course she had felt obliged to take. She was *very* moved to discover how impressively powerful she was in discerning orderliness in the complex mess we call language. All she could articulate was,"I didn't know I knew so much!" She was referring to her knowledge—automatic and unthinking and, for that reason, discounted as simple and unimportant—of how to construct the plural of an English noun. Although she spoke only of what she "knew," it was clear that her newfound respect included *herself as thinker* as well as the knowledge she housed.

How did she come to recognize her own knowledge and theory-making ability? She had to teach what she knew to our computer, which did not know how to construct plurals of English nouns. Because the programming was simple enough, she could attend to the linguistic details.

Those details are remarkably *not* simple. A first problem is how to teach the computer explicit rules for what she herself did so automatically. Of course, in the sense that she could instantly generate any number of words and their plurals, it could be said that she already "knew" those rules. But being able to use the system and being able to articulate its rules are quite different. She treated the "adding *s*" rule as trivial, failed to notice that she ever added *es*, and was *aware* of only one special rule, one that she thought she had learned in elementary school: "if the word ends in a *y*, change it to an *i* and add *es*." Like the once official line on protons, this rule is wrong, and she discovered it when she taught it to her program and it then faithfully converted *boy* into *boies*.

Two roles for the computer: stupidity and good management

But why drag in the computer? Could her investigation not have proceeded just as well without the computer? I think not. In part, it is difficult to test out ideas unless one has a slave stupid enough *not* to help. The computer is just that. Because my student was already fluent with her language and knew how to make the plural of the word *boy* she didn't think of it as "a word that ended with a y" but rather just as the word *boy*. Her discovery of *boies* was an accident. She was showing me her "finished" plural machine and typed the word *boy* in (perhaps because *boy* was not infrequently on this senior girl's mind?) and there it was: *boies*. Although she could easily have been led to say that there are words that require *es* for their plurals, she rediscovered that also in the course of trying a rule that knew only about adding *s*.

Perhaps most importantly, the computer helped by letting her deal with the problem in manageable parts. Language is complex and those of us who are not trained linguists do not, in general, have the concepts and techniques to manage its complexities without help. Here is a route not unlike the one my student took, toward the solution of the plural problem. Her first cut at the rule was something like this:

```
TO PLURAL :NOUN
        To build the plural of any particular noun,
 OUTPUT WORD :NOUN "S
    just output a WORD consisting of the NOUN and the
    letter S.
END
```

A next step might acknowledge that words that end in *s* add *es* to make their plurals:

```
TO PLURAL :NOUN
 IF "S = LAST :NOUN → [OUTPUT WORD
    :NOUN  "ES]                          If s is the last
    letter, then use es in the output instead of just s.
 OUTPUT WORD :NOUN "S
                      Otherwise, proceed as before.
END
```

The student might then find other spellings that require *es* to make their plurals, and can add these, one at a time, as they are discovered.

```
TO PLURAL :NOUN
 IF "S = LAST :NOUN [OUTPUT WORD
    :NOUN "ES]
 IF "O = LAST :NOUN [OUTPUT WORD
    :NOUN "ES]     If o is the last letter, use es.
 IF "X = LAST :NOUN [OUTPUT WORD
     :NOUN "ES]       If x is the last letter, use es.
 OUTPUT WORD :NOUN "S
END
```

Of course, there are more "elegant" programming statements that avoid the need for the three nearly identical lines in the program shown above. There is a stage at which the drive for elegance helps students identify underlying patterns, but initially I prefer to keep the programming in the background, so attention is not drawn away from the real issue, which is the discovery and articulation of linguistic rules.

Often, along the way, the student would recognize a whole class of "special cases" like words ending with *y*. There are many ways of doing the programming, but my interest as a teacher is now on the linguistics—the science of the matter—and not the computer particulars, and so I encourage handling the new class by itself. As a result, the programming remains familiar, and again avoids drawing undue attention to itself. So the student might build a highly specialized rule to deal correctly with words that end with the letter *y*, but adapted only for those words.

```
TO YPLU :NOUN
 IF VOWELP LAST BUTLAST :NOUN →
   [OUTPUT WORD :NOUN "S]
 OUTPUT WORD (BUTLAST :NOUN) "IES
END
            To form the plural of a word that ends with the
                                                 letter y,
                          if the next-to-last letter is a vowel,
                          just add an S to the end of the word.
          Otherwise, remove the last letter (Y), and add IES.
```

The commands PRINT YPLU "BOY and PRINT YPLU "LADY will do the right thing, but because MOUSE does not end in *y*, PRINT YPLU "MOUSE will print MOUSIES.

Having built a specialist, the student might wish to incorporate it into the original procedure, perhaps with a statement like this:

```
 IF "Y = LAST :NOUN →
                              If the last letter is y,
   [OUTPUT YPLU :NOUN]
           let the new specialist YPLU handle it.
```

By alternating between partial models and experiments to determine how well these models work, the student may incrementally build up a very complex rule.

```
TO PLURAL :NOUN
 IF MEMBERP :NOUN [SHEEP DEER etc.]
    [OP :NOUN]
               Handle uninflected plurals properly.
 IF :NOUN = "CHILD [OP "CHILDREN]
                              Handle a true exeption.
 IF MEMBERP LAST :NOUN [S O X] [OP WORD
    :NOUN "ES]                Handle the es cases.
 IF "Y = LAST :NOUN [OP YPLU :NOUN]
 IF "H = LAST :NOUN [OP HPLU :NOUN]
          Handle rush, catch, wrath, catarrh, etc.
 IF COMPARE.END [F FE] :NOUN [OP FPLU :NOUN]
                         Handle half, knife, muff, etc.
 OP WORD :NOUN "S
END
```

Sometimes, in order to keep the programming simple, I provide tools not normally found in the language. COMPARE.END is one such tool, which allows a student to compare the end of a word with a set of possible endingswithout having to build the machinery to do so.[1] In this example, the student uses COMPARE.END with the list [F FE] in order to find words like *half*, *knife*, *muff*, and so on. This student was already familiar, from earlier projects, with other ways of comparing parts of words (MEMBERP and = LAST in this example), or I might otherwise have begun with COMPARE.END from the start, leading to an even simpler program such as this:

```
TO PLURAL :NOUN
 IF COMPARE.END [SHEEP DEER etc.]
    :NOUN [OP :NOUN]
                         Works even for mousedeer!
 IF COMPARE.END [CHILD]
    :NOUN [OP "CHILDREN]
                         Works even for brainchild!
 IF COMPARE.END [S O X]
    :NOUN [OP WORD :NOUN "ES]
 IF COMPARE.END [Y]
    :NOUN [OP YPLU :NOUN]
 IF COMPARE.END [H]
    :NOUN [OP HPLU :NOUN]
 IF COMPARE.END [F FE]
    :NOUN [OP FPLU :NOUN]
 OP WORD :NOUN "S
END
```

I provided this lengthy example in order to make two points.

The first deals with access to linguistic study. A complete rule for making plurals can be quite complex, but when one has adequate tools, each step along the way is simple to articulate and add to the description. If the description had remained on paper, there would come a point when it became too complicated and dry for non-linguists to check. On the other hand, when the description is a computer program, the computer can do the checking.

The second is also a consequence of using the computer to do the checking. The fact that every hypothesized rule for plurals can easily be checked motivates experimentation. Students learn how to choose their experiments because they are actually seeking answers. Finding a way of defeating their rule becomes not a sign of failure, but an opportunity to improve the rule. This, too, is good science.

---

[1] An appendix to this article gives the text of the two special-purpose tools used in these investigations.

Modeling versus describing: introducing a new concept

Another important contribution that the computer makes was brought to my attention by a colleague who referred to the computer program as an "unambiguous, externalized model-as-conjecture" that may help to highlight the very *idea* of modeling.[2] As unaccustomed as students may be to giving accurate and detailed descriptions of anything, they are very unused to thinking about any such description is an implied model—an abstraction of presumably important details from presumably superficial or irrelevant ones. At one level, a computer program is merely an accurate and detailed description. But programs are not just reading matter; they are intended to cause computers to exhibit behaviors. The impact of that difference is observable every time a new group of students first models gossip.

When I first introduce the idea of modeling to a new group—whether the group is fourth graders or college students—I do not begin with projects like PLURAL which investigate the microstructure of words but with the far more accessible realm of sentence structure. Even that domain is too broad, so I constrain the kind of sentence that the group will deal with by declaring that we will try to model gossip. Gossip has many advantages: it is funny, personally engaging (even if the group chooses no *real* gossip!), and sufficiently stereotyped in form that it is an easy place to begin. I start by asking people to generate data by offering some of their own examples of gossip. At a recent graduate level class for teachers, the first sentences I got (by three different participants) were these:

I heard that the ROTC students had to
skin rabbits for survival skills.
I heard that Fred kissed Wilma.
I heard that the business teacher snuck
into the janitor's room to smoke.

A first approximation model of these sentences takes note only of the larger structural elements. Each sentence, as it turns out, begins with the words I heard that, then states who the gossip is about (Fred, the ROTC students, the business teacher), states what that subject did (had to skin rabbits, kissed Wilma, snuck into the janitor's room), and then contains a third optional element explaining the purpose of the act (for survival skills, to smoke). Although this analysis is every bit as much a model as the program we are about to write, it does not seem to *function* as a model for beginners. It feels, to the non-linguist, adequately descriptive but does not seem to suggest generative possibilities.

But then we build the model. When we type PRINT GOSSIP we want the computer to produce a sentence that is gossipy. So we teach the computer to gossip by outputting a sentence consisting of four parts: a verbal formula we shall call INTRO, a part that says WHO, a DIDWHAT part, and a WHY part. We encode our description this way:

```
TO GOSSIP
 OUTPUT (SENTENCE INTRO WHO DIDWHAT WHY)
END
```

The WHO part must pick among the available options and output the result. It looks like this:

```
TO WHO
 OUTPUT PICK [Fred [the ROTC students]
    [the business teacher]]
END
```

Similarly, DIDWHAT selects and outputs from its own list of words or phrases:

```
TO DIDWHAT
 OUTPUT PICK [[had to skin rabbits] [kissed
    Wilma] → [snuck into the janitor's
    room]]
END
```

The INTRO offers (in our sample) no options, and might simply be written this way:

```
TO INTRO
 OUTPUT [I heard that]
END
```

but it might equally well be written as if it were PICKing from a list of only one phrase, the choice being mostly a matter of pedagogical style.

```
TO INTRO
 OUTPUT PICK [[I heard that]]
END
```

Finally, the WHY must contain a "no-word phrase" to allow GOSSIP to create the unexplained Fred kissed Wilma sentence.

```
TO WHY
 OP PICK [[for survival skills]
    [to smoke] [ ]]
END
```

The difference between description and modeling becomes apparent as soon as this model is run. Students of all ages, who have followed or even helped to perform the analysis, have created the procedures, and have presumably "understood" the description are still almost always surprised and delighted when the command PRINT GOSSIP results in a sentence like *I heard that the business teacher kissed Wilma for survival skills.* People are surprised at how humanly a computer and a bunch of mathematical rules can behave. They are impressed with their own ability to control that behavior. They are almost always engaged. And if they add their own names to the list and produce a WHO DIDWHAT WHO WHY model, they find the results even funnier and more engaging. And, while students are rarely motivated to tinker with a *description* they readily begin to tinker with the *model*—a critical element in scientific and mathematical creativity.

---

[2] Andee Rubin, BBN Laboratories, Cambridge, MA, personal communication.

Here are two examples of tinkering that require no new programming concepts:

```
TO QUESTION
 OUTPUT (SENTENCE [Was it] WHO [who] DIDWHAT
    [with] WHO [?])
END

TO CONDITION
 OUTPUT (SENTENCE [I don't know who] DIDWHAT
    WHY  →  [but if] WHO DIDWHAT [,] WHO
    [will be very mad.])
END
```

By the way, it is a considerable benefit to the modeling process, but not at all a coincidence, that the three sentences we began with in this example have such similar structure. The first sentence volunteered by a member of a group tends to set a pattern which other members unconsciously follow. This is particularly reliable among children, who tend to produce simpler constructions anyway, like "Dana loves Lee" or "Rudy hates Sandy." But even adults, who often try deliberately to be inventive with form, tend to embellish rather than alter the model set in the first sentence. Another way of saying this is that the modeling part is natural: people model their behavior (their suggested sentences) on the behavior they see. What we are teaching them is not how to *make* models, but how to do so consciously and how to articulate what they know.

Science as real inquiry

Of course, if an entire group of students is asked to investigate the same aspect of the same phenomenon, the experience can quickly shift from one of discovery to one of exercise—a dreary and unsatisfactory substitute. So, after introducing the requisite techniques for exploring words (e.g., with a project like PLURAL) or sentences (e.g., with GOSSIP), it is important to suggest other explorations that the same programming skills make possible. For example, when students are thinking about how to make plurals, they may also explore adding other affixes—suffixes other than *-s*, and prefixes.such as *in-* or *ex-*. A very small variation in programming allows students to investigate these problems *phonologically* rather than *orthographically*. With but a few technical tools, one can thus foster a research atmosphere in which students explore a variety of related phenomena and then pool their discoveries to collaborate in developing deeper insight into the phenomena.

With the same programming techniques, students can also begin to explore spelling: another entire class of rules one already knows but cannot always articulate. Nobody has been taught a rule for spelling the *ow*-as-in*cow* sound, but when we see how *very* strange PLOWGH, COU, or FLOUER look, it seems like more than memorization of each individual word is at work. In fact, if you make up nonsense words with the *ow*-as-in*cow* sound in them, the chances are good that you will spell them "correctly"—that is, in accord with the rule that governs the vast majority of English words that contain this sound. The modeling task is to write a program that takes, as inputs, the letters before and after the *ow*-as-in*cow* sound and then spells the word correctly, like this:

```
?PRINT SPELL "CL "N
CLOWN
?PRINT SPELL "CL "D
CLOUD
?PRINT SPELL "   "L
OWL
?PRINT SPELL "PL "
PLOW
?PRINT SPELL "PL "GH
PLOUGH
?PRINT SPELL "FL "ER
FLOWER
```

Such a rule might begin this way:

```
TO SPELL :HEAD :TAIL
 IF EMPTYP :TAIL [OUTPUT WORD
    :HEAD "OW] OUTPUT (WORD
    :HEAD "OU :TAIL)
END
```

Again, to improve this simple rule, we may proceed one single step at a time. What words does this rule get wrong? What *one* addition might you make to this rule to solve *one* of the patterns that it now gets wrong? What's another...? This incremental approach renders English spelling—often used as the ultimate example of disorderliness—accessible to scientific inquiry. And by non-linguists! Moreover, the incremental approach is a good scientific method.

There are yet other intriguing areas of linguistic inquiry. My students are well aware—if for no better reason than the threat of college boards—that they do not know every word in the English language. Yet, they have a surprisingly good notion of whether some unfamiliar combination of letters is *perhaps* a word that they don't know or whether, instead, it is unlikely to be English at all. So what is the rule that governs what is acceptable as an English spelling? Why might we accept *clonch* as native but not *pribatuek* or *grzyb*?

Remarkably, no new programming concepts are needed. If one has modeled gossip, then one already knows how to make procedures that build linguistic structures from a supply of available units. In this case, the structures to be investigated are words, and the linguistic units are letters, letter combinations, or sounds. Again, one might begin by taking some data from which to build a preliminary model. From the words pounce, stink, clang, brush, mop, and dead, we might create a two-part model consisting of the first "half" and second "half" of each word (divided according to some rule we invent) or we might create the following more complex model:

```
TO BEGINNING
 OP PICK [ST P CL BR M D]
END

TO VOWEL
 OP PICK [A I O OU U EA]
END

TO ENDING
 OP PICK [NG NK P SH NCE D]
END
```

Just as the procedures that picked the elements of gossip were virtually identical in their structure, so are these; they can be assembled as if from a cookbook. That is important because we want the programming to remain in the background. The only thing that differentiates these procedures is the language chunks that they contain. And the only programming element that differentiates NEWWORD, below, from GOSSIP is the glue we use to assemble the component parts. The overall structure remains the same. NEWWORD uses the same glue that PLURAL does.

```
TO NEWWORD
                    To build a new word from pieces,
 OUTPUT (WORD BEGINNING VOWEL ENDING)
        use WORD to glue the components together,
END                       and output the result.
```

This model of English one-syllable words creates real words like cloud, good fake words like brance, but also some words like broung and peank that, although they are pronounceable, seem like clinkers. Students may also, of course, model other languages that they know. The important elements are the modeling and the asking of real questions. *Can* I make a good model of Spanish verbs? And if so, what are the structural elements? How can I fix NEWWORD so that it does better, and what is a good measure of "better"?

What about "real" science and mathematics?

Although my real purpose is to state that inquiry—not the results of someone else's inquiry—should be the object of science education, I have gone further to make the case that language can and ought to be made an important object of that inquiry. Among other reasons, I cite the accessibility of its data, and its humanness. I even propose that there are times when it should supplant the conventional topics of scientific inquiry, primarily as a way to change the perception of science as dealing only with the distant and unknowable. That sense of unknowability can be fostered even when science's *purpose* feels warm and human and close to home, as is the case with medical science, for a single example; its *results* hit home, but its *processes*, except for the trivial, may feel utterly inaccessible.

But, of course, I do not mean that the stars and rocks and electrons and chemicals and cells are *never* to be studied. For one thing, we can enjoy reading about the detective work of a good scientist whose experiments we cannot reproduce. There may also be ways in which computer simulations of some of these domains—microworlds of various sorts—help make them more immediately accessible for exploration.

And mathematics has suffered from some of the same kind of image problem that plagues science. The vitality of mathematics, as mathematicians experience it, is in sharp contrast to its public image. As it is commonly perceived, mathematics is the least creative of subjects: a dead, unchanging body of facts and techniques handed down from the ancients and tolerating no room for inquiry, every question bearing one and only one answer, an answer that is already known by someone. Computer-supported experimentation has changed the face of mathematical research. The opportunity to perform mathematical experiments can be as valuable to students as it is to mathematicians: it opens up previously inaccessible mathematical domains; it provides, through its concreteness, a scaffold for reasoning and a matrix for problem-posing; and it fosters the development and use of qualitative, visually based reasoning styles to augment the traditionally taught symbolic-deductive methods (see, for example, Goldenberg, 1988; Schwartz & Yerushalmy, in press; and Yerushalmy and Chazan, 1987). Two forthcoming secondary- and post-secondary mathematics texts make exemplary use of such exploration and computational modeling approaches (Lewis, in press; Cuoco, in press). A third takes on a piece of 20th century mathematics as its domain (Goldenberg, in preparation).

Elaborations of the language exploration can be found in Goldenberg and Feurzeig (1988) and Sharples (1966). Goldenberg and Feurzeig bring together the philosophy of scientific inquiry and mathematical modeling, and the perspectives of linguistics, language arts teaching, computer science, and cognitive psychology. Written to an adult audience, this book presents most of the content of an introductory college linguistics course, yet many of the suggested explorations, presented somewhat differently, have been performed by students in high-school or elementary school. The book suggests activities with prefixes and suffixes that lead to investigations of spelling and meaning, explorations of the relationship between English and Italian or German cognates, ways to analyze a foreign language, and more. Sharples (1986) presents activities in which young students explore, among other things, their own vocabulary and how they use it in written composition. Its focus is more on language than on science or modeling, but its use of experimentation gives it a somewhat similar spirit.

Because language is so personal, so accessible, and yet so complex, it can become a very engaging object of scientific exploration at many different levels, benefitting one's understanding of the methods of science, the structure of language, and even the nature of mind. Because computer modeling makes the structural elements of such typically abstract subjects as language and mathematics more "concrete" and manipulable, it encourages some of the best features of scientific thinking: inquiry, experimentation, articulation of theories, and the building and refining of models.

Appendix: Special-purpose tools

Two special purpose tools, compare.end and pick, are needed for these studies. I treat these tools as if they were "native" to the language and therefore explain only what they do and how to use them. It is a distraction to bother the student with the details of how these tools have been constructed, just as it would be an irrelevancy to explain how the PRINT command figures out which pixels to light up on the screen.

```
TO COMPARE.END :LIST :WORD
 IF EMPTYP :LIST [OP "FALSE]
 IF COMPARE.END.1 FIRST :LIST :WORD
    [OP "TRUE]
 OP COMPARE.END BF :LIST :WORD
END

TO COMPARE.END.1 :ENDING :WORD
 IF EMPTYP :ENDING [OP "TRUE]
 IF EMPTYP :WORD [OP "FALSE]
 IF NOT EQUALP LAST :ENDING LAST :WORD
    [OP "FALSE]
 OP COMPARE.END.1 BL :ENDING BL :WORD
END

TO PICK :OBJ
 OP ITEM (1 + RANDCM COUNT :OBJ) :OBJ
END
```

References

Cuoco, A., in press (expected publication date, fall 1989). Tentative title, *Investigations in Algebra: Combinatorics and Number Theory*. Cambridge, Mass.: The MIT Press.

Goldenberg, E. P. *Fractals and other curves: exploring a visual mathematics*, in preparation.

Goldenberg, E. P. Mathematics, Metaphors, and Human Factors. *J. Mathematical Behavior*, 7, pp. 135-173. September, 1988.

Goldenberg, E. P., and W. Feurzeig. 1987. *Exploring Language with Logo*. Cambridge, Mass.: The MIT Press.

Lewis, P. G., in press (expected publication date, fall 1989). Tentative title, *Exploring Precalculus Mathematics Discretely: A Computer Laboratory Approach*. Cambridge, Mass.: The MIT Press.

Schwartz, Judah L., and Michal Yerushalmy. The *Geometric Supposer*: using microcomputers to restore invention to the learning of mathematics. In *Contributors to thinking*. Eds. Perkins, D., Lochhead, J., and Butler, P. Hillsdale, NJ: Erlbaum (in press).

Sharples, M. 1986. *Cognition, Computers, and Creative Writing*. Chichester, England: Ellis Horwood; Distributed in U. S. by J. Wiley.

Yerushalmy, Michal & Daniel Chazan. Effective problem posing in an inquiry environment: a case study using the *Geometric Supposer*. *Proceedings of the Eleventh International Conference on Psychology of Mathematics Education*, vol. 2., pp. 53-60. Eds. Bergeron, Herscovics, Kieran. Montreal, 1987.

USING HYPERCARD IN A FIRST COLLEGE COURSE IN COMPUTING

David E. Cummins
Department of Computer Science
Lewis & Clark College
Portland OR 97219

James R. Ullrich
Department of Computer Science
University of Montana
Missoula MT 59812

Abstract

The authors relate their experience based upon the experimental use of HyperCard in a first college level computer course. All things considered the experiment was judged quite successful. By the end of the course each student had carried out a significant HyperCard project related to his or her individual area of interest. Furthermore HyperCard provided a natural path from programming-free computing, through examining existing programming behind various HyperCard facilities, to the writing of simple and sometimes not so simple HyperTalk scripts. HyperCard was flexible enough to accomodate the varied aptitudes and experience levels of the class members, an important attribute in an introductory class. The authors offer a number of observations concerning some of the assets and liabilities of HyperCard in a classroom setting.

Background

Apple's HyperCard has been getting a lot of attention within the educational community. One reason for this is HyperCard's potential as an authoring tool (Campbell, 1988). Still another is the claim that HyperCard represents a breakthrough in the quest for a truly natural and intuitive human-machine interface for "the rest of us", i.e. nontechnical and perhaps inexperienced computer users. In its promotional literature Apple states, "Now ordinary people can sit down in front of a computer and make it do precisely what they want it to do". They further describe HyperCard as "a tool for the mind -- one tool that bends to every individual purpose". HyperBole (pun intended) aside, reviews (e.g. Williams, 1987, Dear, 1988) have generally supported the assertion that HyperCard does provide a reasonably natural way to use a computer to represent, organize and access information, provided it is used within an appropriate domain of applicability.

HyperCard in the Introductory Computing Course

It occurred to the authors that HyperCard might turn out to be an ideal tool for use in a college level introductory computing course taught in a liberal arts college. The first author pieced together the course and offered an experimental section using HyperCard in January 1988. The outcome was generally positive, and the course was polished with help from the second author and offered again in the following quarter. The second offering corroborated the positive impressions of the first experience. While no formal measures of educational efficacy were attempted, the projects turned in by the students and the ambition evidenced through these projects provide impressive testimony to the promise of HyperCard in the classroom.

Perhaps some words about the philosophy of the intodunctory computing course at Lewis & Clark College will be helpful in describing the context of the HyperCard experiment. The introductory course is taught by computer science faculty as a service to the institution. Within the last four or five years the course has been taught in a number of ways. Early commitment to a conceptual course emphasizing an overview of the science of computer science -- with an emphasis on algorithms but not necessarily programming -- gradually gave way to a much stronger emphasis on software tools, in particular spreadsheet and database packages. Throughout the evolution of the course persistent efforts were made to embed whatever specific skills were being taught within a conceptual framework. This was the computer science signature of the course.

The HyperCard experiment represented no departure from this tradition. Before HyperCard was introduced, considerable material was presented on what a computer is and is not, what information is and how it can be represented within a computer, how human cleverness can be encapsulated in the form of algorithms and programs, and why computers are changing in some very fundamental ways the very fabric of contempory society. Furthermore, as the facilities of HyperCard were presented, the processes which underlay what students saw on the screen were always explained. Students discovered, for instance, why -- if they were to count bits -- most pictures are at least as voluminous as a thousand words. Subsequently they were not terribly surprised when graphics intensive stacks took up so much space on their diskettes, data compression notwithstanding.

A question uppermost in the authors' minds -- above and beyond the issues of whether HyperCard would be easy to learn, exciting, motivating, etc. -- was whether HyperCard, with all of its particulars about creating buttons, cloning fields, etc. might impede efforts to maintain a meaningful conceptual focus within the course. We found the answer to be no. HyperCard presented many opportunities for presenting interesting aspects of computer science, not the least of which were concepts of data encapsulation and object oriented programming. In addition HyperCard -- with its scripting (a euphemism for programming) underlying every action -- provided a readily available mechanism for transition from the world of buttons and mice to the world of algorithms. Whether a student made deep forays into HyperTalk (HyperCard's programming language) or nervously probed its periphery -- we were delighted when we observed the former but understood the latter -- was very much an individual decision. Thus HyperCard offered plentiful opportunity for "deep" exploration, yet was always available at whatever level an individual student opted for. This tolerance for individual differences turned out to be the most important single factor in HyperCard's success in the classroom environment.

Helping HyperCard to succeed did require more than a little effort on the part of the course designers. Textbooks with class exercises, homework assignments, etc. were nowhere to be found. The Apple manuals were not designed to support classroom instruction. The only book available at the time was a paperback (Goodman, 1987) in the genre of the many "Inside [your favorite microcomputer software package]" books found in computer stores. Its coverage of HyperCard was quite complete, but not always well paced, and it offered limited conceptual content. Devising class exercises and examples which were graduated in difficulty turned out to be particularly challenging. Excursions through numerous collections of "stackware" yielded examples that for the most part were much too complex for our purposes. Simple but informative examples were hard to come by. We found ourselves creating many of our own resources, augmented by distillation of certain interesting components of HyperCard stacks we happened to come across. HyperCard is a system in which it is very easy to become mired in detail, and lose sight of the forest for the trees.

Assets and Liabilities

At this point perhaps the most useful thing we can do is set forth what we found to be the assets and liabilities of using HyperCard in an introductory computer class. In our minds the assets clearly outweigh the liabilities, but the latter are not trivial, and should serve as caveats to anyone considering a course of action similar to ours.

Some attributes we found to be assets in the classroom environment:

- First and foremost, HyperCard adjusts well to individual differences in aptitude, experience and interest found in students in an introductory class. For example, some students might choose to invest their energy in exploring the finer points of scripting at the expense of time spent in mastering graphical rendering techniques, while others might do just the reverse.
- HyperCard provides for a transition from an initial reliance on high level mechanisms, such as invoking some action by clicking on a button, to designing links between buttons and actions without programming, to creating more flexible and more complex links with programming if desired.
- A corollary to the above observation is that students discover for themselves a rather deep fact about the relationship between computers and the people who use them. Programming in the traditional sense is often not necessary when using a computer, but there is nevertheless very complex programming behind what is actually a virtual computer that the user is dealing with. Furthermore, if an individual wishes or requires increased fluidity of function and customized behavior in that virtual computer, then programming will usually be required. This is true and delightfully obvious in HyperCard.
- The flexibility of HyperCard allows students to employ the computer to do something worthwhile within their own domain of interest. Biology majors are likely to explore HyperCard's ability to model some aspect of flora, fauna, or perhaps the structure of DNA. Political science majors might map voting patterns within a given geographical area. Art majors are likely to explore the design possibilities of HyperCard's bit-mapped graphics.
- The integration of graphical information with words and numbers is a fundamental source of the interest and representational power of HyperCard. Sometimes a picture is worth a thousand words.
- The productions (final projects) of some of the students are just plain impressive. A description of three of them concludes the article.
- Lastly, HyperCard is relatively easy to learn, exciting, and highly motivating. We believe that most students would affirm that it is just more fun to work with HyperCard than to enter a budget into a spreadsheet or to maintain names and addresses in a database.

Some attributes we found to be liabilities in a classroom environment:

- The superficial simplicity of HyperTalk, the programming facility of HyperCard, masks considerable underlying complexity. The more ambitious students tend to get into trouble in two ways. First HyperTalk, with its ubiquitous "goto" command, often seems to elicit the very kind of convoluted logic that computer scientists have railed against for three decades. The second area of difficulty involves students running afoul of HyperCard's rather complex rules for generating system messages. Even the rules for mouseUp are daunting. We found it necessary to monitor carefully what students were doing, and to redirect their efforts whenever they were courting frustration.
- The numerical capabilities of HyperCard are somewhat limited. While it is possible to create spreadsheet-like applications in HyperCard, it is usually tedious and awkward to do so. Yet we feel the spreadsheet paradigm is one with which students should become familiar. We have considered using both HyperCard and a spreadsheet package in the introductory course, but we are reluctant to undertake the introduction of two comprehensive software systems in a single course. Perhaps we are being blindsided by our depth rather than breadth approach to computers.
- A last caveat involves our feeling that it is easy to get caught up in the cuteness of HyperCard.

Instructors need to give some thought to what is essential and informative, and what is fluff. We will refrain from imposing our opinion on others, and only suggest that some reflection is in order.

Student Projects

We conclude with brief descriptions of three of the more interesting course projects, each the work of a single student. The projects provide some insight into what can be done by individuals taking their first college course in computers.

- A biology student created a very nice application which enables a naive user to correctly classify any conifer native to the northwestern U.S. The user is guided step by step through criteria such as needle shape, clustering patterns, habitat, etc. -- all supported by very nice graphics -- until a correct classification is made.

- A music major created a menu based system which allows the user to enter a sequence of notes using a representation of a piano keyboard, play them, and revise the score in a very interactive way.

- A student created a preview of all the dormitories at Lewis & Clark, providing information concerning size, traditions, characterizations of typical residents, and the reputation of each dorm as seen from the students' viewpoint.

References

Campbell, Tom. "Macintosh in instruction and research". *T.H.E. Journal*, Macintosh special issue, 1988, 49-50.

Dear, Brian L. "HyperCard: What is it?". *Byte*, 13(8), August 1988, Mac special edition, 71-75.

Goodman, Danny. *The complete HyperCard handbook.* New York: Bantam Books, 1987.

Williams, Greg. "HyperCard: First impressions". *Byte*, 12(14), December, 1987, 109-117.

FUNCTIONAL PROGRAMMING THROUGH EXPRESSION TEMPLATES: USING EXTENSIBILITY AND DYNAMIC SCOPE TO CUSTOMIZE CONTROL STRUCTURES IN LOGO

Brian Harvey
Computer Science Division
University of California, Berkeley

Abstract

Functional programming methodology has made possible a revolution in undergraduate computer science education. Abelson and Sussman's pioneering ***Structure and Interpretation of Computer Programs*** course uses the Scheme programming language as a vehicle to present such wide-ranging, powerful ideas as object-oriented programming, logic programming, and compiler construction to college freshmen. However, the same central ideas that give Scheme its power---first-class functions and lexical scope---can make this course too demanding for students who are not computer science specialists. There is a need to make functional programming ideas available to beginning students whose major interest is in some other field.

Logo, a functional programming language designed specifically for educational use, suggests itself as an obvious vehicle for such a beginning course. Its data structures lend themselves to projects in symbolic computation, so that it can appeal to students with limited background in formal mathematics. But in fact Logo has not been widely used beyond the primary school level. Why not? The reasons are partly accidents of history, but another answer is that Logo's reliance on recursion as its only control mechanism creates too great a barrier that must be crossed before any interesting programs can be written. Luckily, Logo is *extensible* teachers can provide new control structures, implemented in Logo itself. This paper presents a family of such functional programming structures, and describes how they have been used in a new introductory undergraduate programming course.

Functional Programming

Traditionally, the dominant model of a computer program presented in introductory courses has been that of a *sequence* of instructions. The computer carries out the instructions one after another; each instruction has an *effect* such as printing a result on paper or setting the value of a variable. Sequential programming is the only model provided in BASIC, the language most commonly used in secondary education; Pascal, currently the most popular language at the college level, allows other possibilities but still emphasizes sequential programming as the primary style.

Functional programming is an alternative model of computing, embodied in programming languages such as LISP, APL, and Prolog. In the functional model, the basic building-block of a computation is not an instruction with a global effect, but rather an *expression* that produces a *value* as its result. These expressions can be combined not as a sequence of events in time, but through *composition* of functions, in which the value returned by one function provides the argument to another function.

Many computer scientists and educators have concluded that the functional programming model is not merely different, but better. See, for example, Backus [1978], Allen [1984], and Solomon [1986]. My main purpose in this paper is not to contribute to that debate but, taking the functional model as a starting point, to discuss some details of its pedagogic implementation. In this introductory section, however, I shall review some of the arguments for the benefit of readers who may be new to the ideas.

Consider first a very simple problem: multiplying a number by 2. In the sequential model of computing, this calculation might be done for immediate printing:

```
TO DOUBLE1 :NUMBER
PRINT 2 * :NUMBER
END

? DOUBLE1  5
10
?
```

Alternatively, the result of the calculation could be saved as the value of a global variable:

```
TO DOUBLE2 :NUMBER
MAKE "ANSWER 2 * :NUMBER
END

? DOUBLE2  5
? PRINT  :ANSWER
10
?
```

(I am using Logo procedures to embody these ideas, but you can easily translate to your own favorite programming language.) In the functional model, we begin by encapsulating this calculation as a named function, which can then be used for printing, setting variables, etc.:

```
TO DOUBLE :NUMBER
OUTPUT 2 * :NUMBER
END

? PRINT  DOUBLE  5
10
? MAKE  "ANSWER  DOUBLE  7
? PRINT  :ANSWER
14
?
```

But the added power of the functional approach comes from the fact that we can also use the DOUBLE operation as part of a larger computation:

```
? PRINT DOUBLE DOUBLE 5
20
? PRINT 3 + DOUBLE 6
15
?
```

I chose the simple DOUBLE example so that even a reader unfamiliar with Logo programming, or indeed any programming, can follow its details. But the example may be too simple to be convincing; even BASIC allows this sort of numeric-valued algebraic function. Now consider the problem of finding the real root(s) of a quadratic equation. There may be zero, one, or two such roots. A typical program will PRINT an appropriate message for each of these cases. Instead, here is a function that returns a *list* of all the real solutions, however many there may be:

```
TO QUADRATIC :A :B :C
LOCAL [DISCRIM SQRT]
MAKE "DISCRIM (:B * :B) - (4 * :A * :C)
IF :DISCRIM < 0 [OUTPUT []]
IF :DISCRIM = 0 [OUTPUT (LIST -:B/(2*:A))]
MAKE "SQRT SQRT :DISCRIM
OUTPUT LIST (-:B + :SQRT)/(2*:A) (-:B -
    :SQRT)/(2*:A)
END

? SHOW QUADRATIC 2 5 -3
[0.5 -3]
? SHOW QUADRATIC 1 -2 1
[1]
? SHOW QUADRATIC 1 0 1
[]
?
```

If you are given a quadratic equation to solve without any context, this functional version is no better than the more customary approach. However, imagine that you are given a word problem that gives rise to a quadratic equation along with some additional conditions on the result. For example, the unknown may be the length of a line segment, to be determined from some known facts about the area of a figure that includes the segment. If the equation has a positive and a negative solution, only the positive one solves the original problem. A similar situation is one in which the unknown is one side of a triangle and must obey the triangle inequality along with the other, known sides. For such situations, this functional QUADRATIC procedure can be combined with *filtering* functions that will eliminate from the list of solutions any that don't meet the additional conditions.

```
TO POSITIVE :LIST
IF EMPTYP :LIST [OUTPUT []]
IF (FIRST :LIST) > 0 [OUTPUT FPUT FIRST
    :LIST POSITIVE BUTFIRST :LIST]
OUTPUT POSITIVE BUTFIRST :LIST
END

? SHOW POSITIVE QUADRATIC 2 5 -3
[0.5]
?
```

Readers unfamiliar with Logo may find the details of the programming in this example mysterious, but for our purposes all that matters is the result: The QUADRATIC function has been composed with an independently written, general POSITIVE filter function to find all positive solutions to the given equation. This technique would be impossible in BASIC and extremely awkward (requiring explicit pointer variables) in Pascal.

The functional approach is not limited to functions of numbers. For example, here is a function of an English word whose value is the translation of that word to Pig Latin:

```
TO PIGLATIN :WORD
IF (MEMBERP FIRST :WORD [A E I O U]) [OUTPUT
    WORD :WORD "AY]
OUTPUT PIGLATIN WORD BUTFIRST :WORD FIRST
    :WORD
END

? PRINT PIGLATIN "THROUGH
OUGHTHRAY
?
```

Note in passing that this function is defined inductively; if the word begins with a consonant, the value of the function is defined to be the result of applying the same function to another word formed by moving the first letter to the end of the word.

Teaching Functional Programming

At Berkeley we offer two introductory programming courses, one for prospective Computer Science majors and one for students in other departments who want to learn to program either from personal interest or to satisfy a requirement imposed by their own departments. Until three years ago, the two courses were similar in many respects; they were both Pascal-based, both followed the ideas of structured programming, and in fact both used the same textbook [Cooper and Clancy, 1985]. The main differences were that the course for Computer Science majors involved a deeper understanding of large-scale programming methodology and required students to write recursive procedures.

Three years ago the course for Computer Science majors was replaced with a new version, using functional programming, based on the textbook *Structure and Interpretation of Computer Programs* by MIT professors Harold Abelson and Gerald Jay Sussman [1985]. The streamlined syntax of LISP and the wide applicability of functional methodology have allowed this course to go much further than our earlier version in introducing students to exciting ideas at the frontiers of computer science. Weeks that would formerly have gone into the precise placement of semicolons are now devoted to such topics as object-oriented programming, logic programming, constraint systems, interpreters, and compilers.

Until now, however, the course offered for students in other departments has retained its old, Pascal-based curriculum. The Abelson and Sussman course is exciting, but we have considered it too demanding for this second group of students. We are thus in the uncomfortable position of teaching two courses that are rather antagonistic in their intellectual content; students who take both courses, as a few do, find the change in direction confusing. This year we are offering an experimental course that attempts to make functional ideas accessible in a less demanding form.

One of the difficulties in the Abelson and Sussman course, for students without a strong background in mathematics, is its early reliance on recursion as the principal control structure. The first significant programming example in the text (Newton's algorithm for the square root function) uses recursion to express an iterative approximation technique. The authors remark, correctly, that the procedure call mechanism is adequate to express any desired control structure, but they do not comment on the issues of variable scoping that beginners find mysterious in recursion. (That is, when a recursive procedure is invoked we end up with several variables that all have the same name, but different values, existing at once.) Exclusive reliance on recursion in place of more narrowly defined iterative control mechanisms has similarly been cited as a reason not to use Logo in high school programming or math classes [Tinker, 1984; Ropes, 1987].

The central idea of this paper is that functional programming does not stand or fall on the use of recursion as the principal control structure. That is, if you don't want to teach your students recursion in the first week (or ever), you need not therefore retreat to FOR-NEXT programming style. Our new introductory course uses Logo, but with some extensions to the language (easily implemented in Logo itself) to allow functional programming with an iterative style. We are teaching the course on a Vax timesharing system running Unix, using a Logo interpreter written as a summer project by a freshman, Dan Van Blerkom, using techniques he learned in the Abelson and Sussman course. The extensions described below can also be used with the Logo interpreters available for most microcomputers.

Expression Templates

The POSITIVE procedure defined earlier is a typical example of the use of recursion to provide a common functional process, in this case the process of filtering a list according to some condition. The same effect could be written in terms of a general FILTER tool this way:

```
TO POSITIVE :LIST
OUTPUT FILTER [? > 0] :LIST
END
```

FILTER is a procedure with two inputs, a *template* list and a data list. The template is an expression whose value is always TRUE or FALSE, in which a question mark is used to represent the elements of the data list. FILTER evaluates the template once for each element of the data list; its output is the list containing only those input data for which the template is TRUE. In this example the template is the list [?~>~0] and so the output value from FILTER contains only those input data that are greater than zero.

```
? SHOW FILTER [? > 0] [3 27 -5 1.2 0
   6 -4.7]
[3 27 1.2 6]
? SHOW FILTER [MEMBERP ? "AEIOU]
   "ELEPHANT
EEA
?
```

The second example above shows that FILTER, like most Logo operations, will accept a word in place of a data list; in this case the template expression is computed for each letter in the word.

Although FILTER is not a Logo primitive, it has the same syntax as primitive control structures like IF and REPEAT. Logo is an extensible language in which user-defined procedures follow the same notation as the built-in features of the language. This property is in sharp contrast with the situation in languages such as BASIC and Pascal, in which each control structure has its own unique syntax, and the invocation of user-defined procedures follows yet another syntactic rule. The Logo control structures presented in this paper can be taught to students as part of the language, with no special mystique.

The expression template in FILTER is used to select elements of the input data list, but the values computed by the template are not directly present in the value output by FILTER. Another common situation is that a function is to be applied to each input datum, and the values computed by that function constitute the result of the computation. This is called *mapping* a function over a list:

```
? SHOW MAP [? * ?] [1 2 3 4]
[1 4 9 16]
? SHOW MAP [FIRST ?] [EVERY GOOD BOY
   DOES FINE]
[E G B D F]
?
```

As an example of how functional programming could be used in a larger project, suppose we are writing a program to play the game Hangman. The computer has chosen a secret word, and the user is guessing letters. We want to display a partial view of the word, with the correctly guessed letters in their proper positions and the remaining letters represented by underscore characters as placeholders.

```
TO DISPLAY :SECRET :GUESSED
```

```
OUTPUT MAP [IFELSE MEMBERP ? :GUESSED [?]
    ["\_{$\,$}]] :SECRET
END
```

```
? PRINT DISPLAY "POTSTICKER [E T A O
    N]
_OT_T_ _ _E_
?
```

Another common situation is that you want to compose a function with *itself* several times:

```
TO POWER :BASE :EXP
OUTPUT CASCADE :EXP [? * :BASE] 1
END
```

```
? PRINT POWER 3 4
81
?
```

CASCADE takes three inputs. The first, a number, indicates how many times the second, an expression template, should be computed. The question mark in the template represents the result of the previous computation, except for the first time, when it represents the initial value given as CASCADE's third input. So the example above starts with the initial value 1 and multiplies it by the BASE variable the number of times indicated by the EXP variable.

```
TO NTH :INDEX :LIST
OUTPUT FIRST CASCADE :INDEX-1 [BUTFIRST ?]
    :LIST
END
```

```
? PRINT NTH 4 [THIS IS A VERY LONG
    LIST OF WORDS]
VERY
?
```

One advantage of the programming style presented here over the traditional FOR-NEXT iterative style is that it avoids the need for auxiliary index variables. This advantage can be extended from functional computations to iterative, sequential computations by providing control structures based on *instruction* templates rather than *expression* templates:

```
TO PRAISE :FLAVORS
FOREACH :FLAVORS [PRINT SENTENCE [I LOVE] ?]
END
```

```
? PRAISE [GINGER [ULTRA CHOCOLATE]
    [RUM RAISIN] PUMPKIN]
I LOVE GINGER
I LOVE ULTRA CHOCOLATE
I LOVE RUM RAISIN
I LOVE PUMPKIN
?
```

The programming tools shown here are taken from a larger set presented in *Computer Science Logo Style, Volume 3: Advanced Topics* [Harvey, 1987]. They were originally developed for use in specific projects, to reduce the size and increase the clarity of very long program listings. What is new in the course we are offering this year is that these tools are taught from the first day, presented as primitives.

Teaching Experience

The Pascal-based course that we are hoping to replace is taught to about 300 students each semester. We are now teaching the experimental version to a single section of 16 students. None of these students are computer science majors, although a few are freshmen who have not chosen a major and are considering computer science. For some students this is the first experience using computers in any way; others have had prior programming experience in BASIC.

Preliminary results indicate that the provision of functional programming tools does make it possible for students to work on more interesting problems in the early weeks of the course than they could handle with Logo primitives alone. For example, the second homework assignment included the problem of finding perfect numbers by testing each number in a given range. Testing for a perfect number is done by constructing (with CASCADE) a list of all the integers less than the number in question, then filtering out (with FILTER) those that aren't factors, then adding up all the numbers in the resulting list (using REDUCE, another tool in the package).

Logo has a very uniform syntax, in which everything is done by naming a procedure and then supplying expressions for its inputs. The first week of the course is devoted to this model of procedure evaluation. A few students who joined the course after the first week have had trouble with later topics because they missed this crucial idea, but those students who do understand the evaluation model have not, so far, had any trouble with more "difficult" ideas; even recursion, introduced in the fourth week of the course, has not proven difficult for those students who learned thoroughly the ideas presented at the beginning of the course. In future semesters, therefore, we shall emphasize even more strongly the evaluation model.

(The evaluation model embodies, implicitly, the idea of recursion. A Logo instruction consists of a procedure name followed by expressions to provide the inputs to the procedure. Each expression is itself a procedure name followed by expressions for the inputs. It may be that this first glimmering of the idea of recursion makes it easier for students to understand recursive procedure definitions later, but I have no direct evidence for this conjecture.)

The utility of iterative tools does not end once recursion is introduced. Consider the following program, written using only Logo primitives, to generate the leaves of a symmetric decision tree:

```
TO TRAVERSE :TREE
TRAVERSE1 :TREE []
END
```

```
TO TRAVERSE1 :TREE :SOFAR
IF EMPTYP :TREE [PRINT :SOFAR STOP]
TRAVERSE2 (FIRST :TREE) (BUTFIRST :TREE)
    :SOFAR
END

TO TRAVERSE2 :CHOICES :REST :SOFAR
IF EMPTYP :CHOICES [STOP]
TRAVERSE1 :REST (SENTENCE :SOFAR FIRST
    :CHOICES)
TRAVERSE2 (BUTFIRST :CHOICES) :REST :SOFAR
END

? TRAVERSE [[SMALL LARGE] [VANILLA
    CHOCOLATE] [SUNDAE MILKSHAKE]]
SMALL VANILLA SUNDAE
SMALL VANILLA MILKSHAKE
SMALL CHOCOLATE SUNDAE
SMALL CHOCOLATE MILKSHAKE
LARGE VANILLA SUNDAE
LARGE VANILLA MILKSHAKE
LARGE CHOCOLATE SUNDAE
LARGE CHOCOLATE MILKSHAKE
?
```

The structure of this program is hard to follow. [TRAVERSE1 and TRAVERSE2 are mutually recursive. There are two kinds of recursion in TRAVERSE2, one that goes deeper down the tree and one that moves "sideways" through the children of a single node. Compare this equivalent version:

```
TO TRAVERSE :TREE
TRAVERSE1 :TREE []
END

TO TRAVERSE1 :TREE :SOFAR
IF EMPTYP :TREE [PRINT :SOFAR STOP]
FOREACH FIRST :TREE [TRAVERSE1 (BUTFIRST
    :TREE) (SENTENCE :SOFAR ?)]
END
```

While still not trivial, this version does make clear that both an iteration (the FOREACH) and a depth recursion (the invocation of TRAVERSE1 within itself) are involved.

Why Not Use Scheme?

Our goal was to create a course that would be intellectually compatible with the Abelson and Sussman course but less challenging. One obvious approach would be to base the course on Scheme, the dialect of LISP used by Abelson and Sussman. Why did we choose Logo instead? One reason was that we happened to have a suitable textbook available [Harvey, 1985], but there are also introductory Scheme texts [Eisenberg, 1988; Dybvig, 1987].

My feeling is that the very things that give Scheme its exceptional power, compared to other dialects of LISP, also make it a difficult choice for the approach suggested here. These things are first-class procedures and lexical scope.

Scheme extends the principle of functional programming by providing not only numeric-valued functions (like square root) but also functional-valued functions (like derivative or integral). Here is a simple example of a Scheme function whose return value is a function:

```
(DEFINE (MAKE-ADDER A)
   (DEFINE (ADDER X)
      (+ X A))
   ADDER)

> (DEFINE 1+ (MAKE-ADDER 1))
1+
> (DEFINE 3+ (MAKE-ADDER 3))
3+
> (1+ 7)
8
> (3+ 7)
10
> ((MAKE-ADDER 6) 7)
13
>
```

Each invocation of MAKE-ADDER defines a new function, always with the internal name ADDER, but a different function each time. Each of the returned ADDER functions adds some constant to its argument.

In the internal definition of ADDER, the variable X is a formal parameter, like an input in a Logo procedure title line. What about the variable A? In Logo this variable reference would find its value in the environment in which the ADDER function is invoked. (This rule is called *dynamic scope.*) In Scheme, the value comes from the environment in which ADDER]is *defined*---that is, from the MAKE-ADDER environment. (This rule is called *lexical* scope.)

There are many good reasons why most computer scientists prefer lexical scope over dynamic scope. Abelson and Sussman discuss some of those reasons at length in their text. One is that lexical scope provides a mechanism through which several Scheme procedures can share a common local variable. Another is that lexical scope supports "referential transparency": you can tell just by reading a procedure definition what variables it uses, without having to know the context in which the procedure will be invoked.

Lexical scope, though, makes it harder to provide programming tools based on the template model used in this paper. The fundamental problem is that in a lexical language, a procedure is not the same thing as the text of the procedure definition. In the example above, the functions 1+ and 3+ are quite different in effect, and yet they are defined with exactly the same text, the ADDER definition within MAKE-ADDER. The expression template model depends on the fact that in Logo such a template is equivalent to a function.

Of course Scheme can provide tools that are equivalent in effect. The point is that Scheme's versions are not pedagogically equivalent. Instead of

```
MAP [? * ?] [1 2 3 4]
```

Scheme would say

```
(MAP (LAMBDA (X) (* X X)) '(1 2 3 4))
```

The Logo expression template is just a list, like any other Logo list. It can be printed or otherwise manipulated. The Scheme function that is generated by the LAMBDA form above is something more magic. It can't be printed:

```
> (LAMBDA (X) (* X X))
#[procedure]
>
```

Scheme tells us that this is a procedure, but the body of the procedure is hidden within opaque walls. The LAMBDA notation, although more general than the Logo expression template, is a greater stumbling block for a beginning student unfamiliar with functional programming ideas.

A related difficulty is in debugging. When an error condition occurs several levels deep in procedure calls, a dynamically scoped language like Logo can provide a *pause* mechanism that allows the user to issue interactive instructions, as usual, but with access to all of the relevant variables of all active procedures. The Scheme user does not have this capability because instructions entered interactively are not within the defining environment of the procedure in which the error occurred. (Some versions of Scheme provide special debugging tools to allow examination of variables, but not using standard Scheme instructions and not with the full functional composition power of the language.)

In making this argument I am not trying to set up an antagonism between Scheme and Logo. On the contrary, I enjoy teaching both languages, and I am suggesting that one can be a useful stepping-stone to the other. The functional programming approach is shared by both languages; Scheme takes the approach further, but Logo may make it more accessible to a novice with minimal mathematical sophistication. My goal is to provide a bridge, in the form of iterative functional programming tools, to help teachers who are more familiar with the sequential model of programming experience the power of functional techniques.

Bibliography

Abelson, Harold, and Gerald Jay Sussman. *Structure and Interpretation of Computer Programs*, MIT Press, 1985.

Allen, John R., Ruth E. Davis, and John F. Johnson. *Thinking About [TLC] Logo*, Holt, Rinehart and Winston, 1984.

Backus, John. "Can Programming be Liberated from the von Neumann Style?" *Communications of the ACM, 21 (8)*:613--641, 1978.

Cooper, Doug, and Michael Clancy. *Oh! Pascal!* Second Edition, Norton, 1985.

Dybvig, R. Kent. *The Scheme Programming Language*, Prentice-Hall, 1987.

Eisenberg, Michael. *Programming in Scheme*, Scientific Press, 1988.

Harvey, Brian. *Computer Science Logo Style*, MIT Press.
Volume 1: *Intermediate Programming*, 1985.
Volume 2: *Projects, Styles, and Techniques*, 1986.
Volume 3: *Advanced Topics*, 1987.

Ropes, George. "BASIC vs.\ Logo: Programming for Algebra," *Council for Logo in Mathematics Education Newsletter, 1* (2), 1987.

Solomon, Cynthia. *Computer Environments for Children*, MIT Press, 1986.

Tinker, Robert. "Logo's Limits: Or Which Language Should We Teach?" *Currents*, 2 (2), Rhode Island College, 1984.

Appendix: Functional Tool Implementation

```
TO MAP :MAP.TEMPLATE :TEMPLATE.LIST
LOCAL "TEMPLATE.VAR
IF EMPTYP :TEMPLATE.LIST [OUTPUT
    :TEMPLATE.LIST]
MAKE "TEMPLATE.VAR FIRST :TEMPLATE.LIST
OUTPUT COMBINE (RUN :MAP.TEMPLATE) (MAP
    :MAP.TEMPLATE BF :TEMPLATE.LIST)
END

TO COMBINE :THIS :THOSE
IF WORDP :THOSE [OUTPUT WORD :THIS :THOSE]
OUTPUT FPUT :THIS :THOSE
END

TO ?
OUTPUT :TEMPLATE.VAR
END

TO ?REST
OUTPUT BF :TEMPLATE.LIST
END

TO FOREACH :TEMPLATE.LIST :FOREACH.TEMPLATE
LOCAL "TEMPLATE.VAR
IF EMPTYP :TEMPLATE.LIST [STOP]
MAKE "TEMPLATE.VAR FIRST :TEMPLATE.LIST
RUN :FOREACH.TEMPLATE
FOREACH BF :TEMPLATE.LIST :FOREACH.TEMPLATE
END
```

```
TO FILTER :FILTER.TEMPLATE :TEMPLATE.LIST
IF EMPTYP :TEMPLATE.LIST [OP
    :TEMPLATE.LIST]
LOCAL "TEMPLATE.VAR
MAKE "TEMPLATE.VAR FIRST :TEMPLATE.LIST
IF RUN :FILTER.TEMPLATE [OP COMBINE (FIRST
    :TEMPLATE.LIST) (FILTER
    :FILTER.TEMPLATE BF :TEMPLATE.LIST)]
OP FILTER :FILTER.TEMPLATE BF
    :TEMPLATE.LIST
END

TO CASCADE :CASCADE.LIMIT :CASCADE.TEMPLATE
    :TEMPLATE.VAR
IF NUMBERP :CASCADE.LIMIT [OP CASCADE1
    :CASCADE.LIMIT :TEMPLATE.VAR]
OP CASCADE2 :TEMPLATE.VAR
END

TO CASCADE1 :CASCADE.NUM :TEMPLATE.VAR
IF EQUALP :CASCADE.NUM 0 [OP :TEMPLATE.VAR]
OP CASCADE1 (:CASCADE.NUM-1) (RUN
    :CASCADE.TEMPLATE)
END

TO #
OP 1 + :CASCADE.LIMIT - :CASCADE.NUM
END

TO CASCADE2 :TEMPLATE.VAR
IF RUN :CASCADE.LIMIT [OP :TEMPLATE.VAR]
OP CASCADE2 (RUN :CASCADE.TEMPLATE)
END
```

THE APPLE CLASSROOM OF TOMORROW

David Dwyer
Apple Classroom of Tomorrow
20525 Mariani Avenue MS: 42C
Cupertino, CA 95014

ABSTRACT

The Apple Classroom of Tomorrow (ACOT) is a research project that explores learning when teachers and students have access to interactive technologies. To do this, ACOT establishes technology-rich classroom sites and encourages teachers to develop new curriculums and methods of instruction that take advantage of the technology. In addition, the project supports both long-term research and research and development efforts.

ACOT R&D combines current learning theories with advanced technologies to produce powerful learning experiences for children. R&D teams typically work with teachers to build curricula, tools, and environments that can be integrated into ACOT and other classrooms. Projects include the development of interactive media tools for writing, an intelligent tutoring system for physics, and hypermedia tools that link procedural and conceptual knowledge in mathematics.

The ACOT program of research includes exploratory and longitudinal studies that examine the potential of technology in high computer access environments. Currently a dozen studies are under way, examining questions such as: How does high access to technology affect curriculum and instruction? How can computers be used to empower students to take responsibility for learning? How do students organize and use information when they have constant access to computers?

The potential for technology in the classroom must be examined over the long term. Ultimately, what we find in ACOT classrooms can tell us a lot about how computers affect the way students learn and teachers teach, giving us an important glimpse of what education can accomplish when technology is readily available.

LEARNING PROBABILITY CONCEPTS AND MODELING WITH COMUTER SIMULATIONS

Mike Perry
Department of Mathematical Sciences
Appalachian State University
Boone, N.C. 28607

Abstract/Introduction

SIM-PAC, "Simulations in Mathematics-Probability and Computing, is a three year project (1987-1990) funded by the National Science Foundation's Materials Research and Development program (grant #MDR 8751110). The project is developing instructional strategies and accompanying materials including microcomputer software for the teaching and learning of probability concepts through the study of computer simulation models. The materials are designed for the advanced secondary level student but may also be useful in undergraduate level instruction. The development and testing phase of this project will be completed in 1988-89 along with a preliminary evaluation of the materials.

The SIM-PAC approach is based on the belief that the learning of probability concepts may be facilitated by the study of computer simulations and simulation model building is an important problem solving approach. The exploration of simulation models is a key ngredient of the instructional strategy.

This paper describes the background and rationale for the project, its goals and objectives, and the instructional strategy utilized by SIM-PAC. It illustrates the nature of the SIM-PAC simulation models and gives examples to illustrate the capabilities of the software as well as how it is used in typical learning activities.

The preliminary evaluation is not available at this time but will be part of the presentation at the NECC. The presentation will include demonstrations of the use of SIM-PAC software including the examples presented herein.

Background and Rationale

Since 1980 a number of reports from professional organizations and special commissions [1,2,3] have called for changes in the U.S. mathematics curriculum. Usiskin [6] summarized the plethora of position statements: "Taken as a body, the reports from inside and outside mathematics education agree almost unanimously that two concurrent changes must take place in secondary school mathematics. First, that the performance of students be upgraded: requirements for graduation should be increased, emphasis should be shifted from rote manipulation to problem solving, and students need to be better at both of these. Second, that the curriculum be updated: estimation, applications, computers, and statistics and probability should play important roles in secondary school mathematics study." NCTM's "Agenda for Action" concluded that "problem solving must be the focus of school mathematics in the 1980's, including the use of the problemsolving capacities of computers to extend traditional problem solving approaches and to implement new strategies of interaction and simulation, and as new technology makes it possible, problems should be presented in more natural settings or in simulations of realistic conditions."

The NCTM Commission on Standards for School Mathematics released a first draft of its Curriculum Standards [4] in October, 1987 which include for grades 9-12: "Students sould develop an intuitive sense of probability in order to make informed observations about the likelihood of an event, as well as to interpret and judge the validity of statistical claims in terms of the underlying probabilistic assumptions. .... Instruction for the majority of students should emphasize the use of simulation techniques and actual experiments. These experiences provide students with a foundation for the formulation and further study of theoretical probability."

The Joint Committee of the American Statistical Association (ASA) and N.C.T.M. began a project in 1984 entitled "A Program to Improve Quantitative Literacy in the Schools" [5]. This project, better known as "The Quantitative Literacy Project" (QLP), has produced curriculum materials to assist teachers in the proper presentation of statistical and probabilistic concepts. These include *The Art and Techniques of Simulation* [7]. The QLP materials are most appropriate for grades 6-9.

SIM-PAC is an extension of the work of the QLP to grades 11-12. The SIM-PAC approach is based on the belief that the learning of probability concepts may be facilitated by the study of computer simulations; simulation model building is an important problem solving approach.

Goals, Objectives, and Materials

The goal of SIM-PAC is to effect changes in secondary mathematics by influencing: (1) curriculum content, (2) teaching styles and the learning environment, (3) teacher perceptions and attitudes, and (4) student perceptions, attitudes, and abilities. Specifically, SIM-PAC (1) recommends what probability concepts and topics are appropriate for the secondary student, and how, where, and at what levels this content can be integrated into the present and future curriculum, (2) provides strategies and activities for teaching and learning probability concepts and problem solving based on the study of computer simulations, and (3) provides strategies and activities for teacher training.

The materials provided by SIM-PAC include (1) student manual, (2) teacher manual, (3) teacher training manual, and (4) software.

Student Population

The intended student population includes the secondary level student in advanced level mathematics or computing classes as well as the gifted and talented at all secondary grade levels.

Learning Objectives

The understanding of probability concepts and the simulation modeling process are the primary learning objectives. Five fundamental probability concepts are emphasized:

(1) Relative frequency interpretation of probability
(2) Random Variable
(3) Expected value of a random variable
(4) Variability of a random variable
(5) Probability distribution of a random variable

The Eight-Step modeling process is described in *The Art and Techniques of Simulation* [7] as follows:

1. State the Problem Clearly
2. Define the Key Components
3. State the Underlying Assumptions
4. Select a Model for a Key Component
5. Define and Conduct a Trial
6. Record the Observations of Interest
7. Repeat Steps 5 & 6 a Large Number of Times
8. Summarize the Information and Draw Conclusions

The Instructional Strategy

A class activity is developed in three phases:

(1) Present a problem;
(2) Develop an appropriate simulation model using the Eight-Step modeling process;
(3) Explore the simulation model.

In Phase 3, SIM-PAC provides ready to use software for the IBM PC or Macintosh microcomputers. SIM-PAC also provides guidelines and outlines for classes in which the students do their own programming.

SIM-PAC provides carefully designed sequences of activities to direct the exploration of models in Phase 3. These activities encourage the student to think in terms of the five fundamental probability concepts. The types of exploration encouraged are:

1. Situation Specific - The model is explored and analyzed for a fixed set of parameters.
2. Parameter Playing - Parameters are changed to study their effect on the behavior of the model.
3. Mathematical Confirmation - A mathematical model (when available) with the same underlying assumptions is compared with the simulation results.
4. Classification of Models - Several specific models are studied and the student is encouraged to recognize that they can all be described by a single more general model.
5. Reexamine the Model - Does the model produce suspicicious results? Should the underlying assumptions be changed?

SIM-PAC Programs

SIM-PAC software is implemented for the IBM-PC or Macintosh microcomputers. With a PC compatible microcomputer (256k) a graphics card is required. The programs were written and compiled with Turbo Pascal (TM) version 4.0. The software has had extensive classroom testing (10 sites). It has been reviewed by teachers and independent reviewers. Thus the program designs are influenced by their intended classroom uses and contain features which facilitate the suggested learning activities.

Descriptions of models

1. Experiment : Roll a set of fair dice

Random Variable : Sum of the faces

2. Experiment : Toss a coin N times

Random Variable : Number of Heads

3. Experiment : Toss a coin until K Heads occur

Random Variable : Number of the toss on which Kth Head occurs

4. Experiment : Spin a spinner until a duplication occurs

Random Variable : Number of spin on which duplication occurs

5. Experiment : There are N types of tokens. Packs of tokens of size K are obtained until at least one of each type is acquired.

Random Variable : Number of packs required to obtain complete set of tokens.

6. Experiment : A coin is tossed N times.

Random Variables : (a) Length of Longest Run of Heads (b) Number of Runs

7. Experiment : A single server waits on a single queue. During each time interval the probability is P1 that a customer will arrive and P2 that a customer will be served (if a customer is waiting). N time intervals pass.

Random Variable : The Longest Length of the queue

8. Experiment : N objects are placed randomly into K boxes. It is desirable to have at least M objects in each box.

Random Variable : The number of boxes that are OK, i.e., contain at least M objects.

9. Experiment : A container is divided into two chambers. N objects are placed in one of the chambers. An object is randomly selected and moved to the other chamber. This movement is continued until equilibrium, i.e., both of the chambers contain the same number of objects (or differ in number contained by one if N odd).

Random Variable : The number of moves required to reach equilibrium.

10. Experiment : Choose a random sample of size n from the population 1,2,...,N

Random Variable : An Estimator of N .

11. Experiment : The population 1,2,...,N has T of its elements tagged.
  I. Choose a simple random sample without replacement of size n from the population, or
  II. Choose elements without replacement from the population until t tagged elements are obtained.

Random Variable : Estimator of N = n*T/t

12. Experiment : A population of objects is classified into "Defective" and "Non-Defective" objects, with proportion P Defective. The population is considered "Good" if P<P0 or "Bad" if P>=P0. Objects are drawn at random from the population until K defective objects are selected or a total of M objects have been drawn. The population size is quite large relative to M. The sample is used to decide if the population is "Good" or "Bad".

The Decision rule is:

> Reject the Population, i.e., decide the Population is "Bad" (P>=P0), if K defective objects are selected within the maximum M draws
>
> Accept the Population, i.e., decide the Population is "Good" (P<P0) if less than K defective objects are selected in the M draws.

Random Variable : 1 if Decision correct; T =0 if Decision incorrect

13. Experiment : A population 1,2,...,N has proportion P of its elements tagged.
  I. Choose a simple random sample without replacement of size n from the population, or
  II. Choose a simple random sample with replacement of size n from the population.

Random Variable : Estimator of P = t/n, t = no. tagged in sample

14. Experiment : A particle takes N random steps of uniform length in one dimension. The probability of a step to the right is p and to the left is 1-p.

Random Variable : The number of times the particle returns to its original position.

15. Experiment : A particle takes random steps of uniform length in one dimension. The probability of a step to the right is p and to the left is 1-p. There are absorbing barriers to the left and to the right of the starting position. The particle steps until it is absorbed by one of these barriers.

Random Variable : Record which barrier the particle reaches first.

16. Experiment : A particle takes random steps of uniform length in two dimensions. The probabilities of a step to the right and left are p1 and 1-p1, up and down are p2 and 1-p2. There are absorbing barriers surrounding starting position. The particle steps until is absorbed by one of these barriers.

Random Variable : The Number of steps

Program Features

Each SIM-PAC program provides a simulation of a random experiment and statistical summaries of at least one random variable. Statistical summaries include graphs, tables, and numerical measures. There is a standard user format and most of the features and capabilities are shared by all of the programs. Each program will allow four active simulations, editing of parameters, zooms of graphs, comparisons of graphs,a summary table for more than four simulations, printer output, and disk storage of data. There are, however, unique features for some of the simulation models.

Illustrations

The following illustrations indicate the nature of student activities, how the SIM-PAC simulation outputs facilitate exploration, and how statistical concepts are experienced in the process. Keep in mind that these are not complete activities.

Illustration 1.

A true-false test requires 60% correct for a passing score. If you make independent guesses on every question of the test what are your chances of passing? Does the number of questions on the test effect your chances of passing?

Appropriate Model: #1

Objectives: Relative variability in the number correct decreases as the number of questions increases. Specifically, the standard deviation of the proportion correct is inversely proportional to the square root of the number of questions.

Exploration involves parameter playing with n=number of questions using the following types of outputs from the SIM-PAC model. Figure 1 shows the active simulation for n=10 after 60 trials. The change in the distribution since 50 trials is highlighted. Figure 2 shows a comparison of the simulated distributions for $n=10, 20, 40, 80$ after 1000 trials with Serial- Mean Plots. This graph emphasizes the convergence of the mean and the scatter of individual outcomes about the mean and demonstrates the shrinking relative variability as n increases. Table 1 gives specific relative frequencies for $n=10$ produced by 1000 trials and Table 2 shows a summary for the four values.

Illustration 2.

Taxi Cabs in a city have serial numbers 1,2,.....,N. You observe a sample of ten taxi numbers: 414, 357, 112, 536, 284, 542, 526, 164, 222, 448. Noting that the largest number observed is 542 we know that there are at least 542 taxis. We could use the maximum number observed in the sample to estimate the total number of taxies! Is this a good estimator? What is its most obvious shortcoming? Can we invent a better estimator? How much would our estimation improve if we observe twenty taxis?

Objectives: The concept of a sampling distribution. Criteria for a "good" estimator: bias and sampling variability. The effect of sample size on bias and variability, and the difference between different estimators when used with the same sample size.

Appropriate Model: #10

Exploration uses parameter playing with n= sample size and the definition of the estimator. This SIM-PAC program provides an interpreter which allows the user to define an estimator in terms of functions such as Mean or Median, Max or Min, and $X(k)$=the kth ordered value in the sample. Figure 3 shows the simulated sampling distributions for the Max estimator using samples of sizes $n=10,20,40,80$. Figure 4 shows the corresponding Serial- Mean plots and Figure 5 gives 90% boxplots for two cases. Each of these conveys the idea of decreasing sampling variability as n increases. The Mean plots and Boxplots show clearly the reduction in bias as n increases. Figure 6 gives comparisons of three different estimators with $n=25$. The definitions of estimators are given in a "dictionary" as shown in Table 3. The summary tables provided (Table 4) give information about all simulations.

Illustration3.

A large shipment of radios has arrived at your warehouse and you must decide if the shipment is acceptable. You are willing to accept the shipment if no more than 10% of the radioes are defective. Up to 50 radios are selected randomly from the shipment. If four defectives are selected the shipment is deemed unacceptable. Is this a "good" decision rule? How could it be improved?

Appropriate Model: #12

Objectives: Concept of error rates in decision making.

Exploration examines the effect on error rates due to changing the parameters of the decision rule. Figure 7 shows the active simulation for a decision rule. The rule is applied to a population with 10% defectives (bad shipment) and to ten populations with percent defectives less than 10% (good shipments). The graphs display the rejection rates and hence the error rates for accepting "bad" shipments and rejecting "good" shipments. Comparisons of two rules for a particular "good" shipment may be made as shown in Figure 8,9.

References

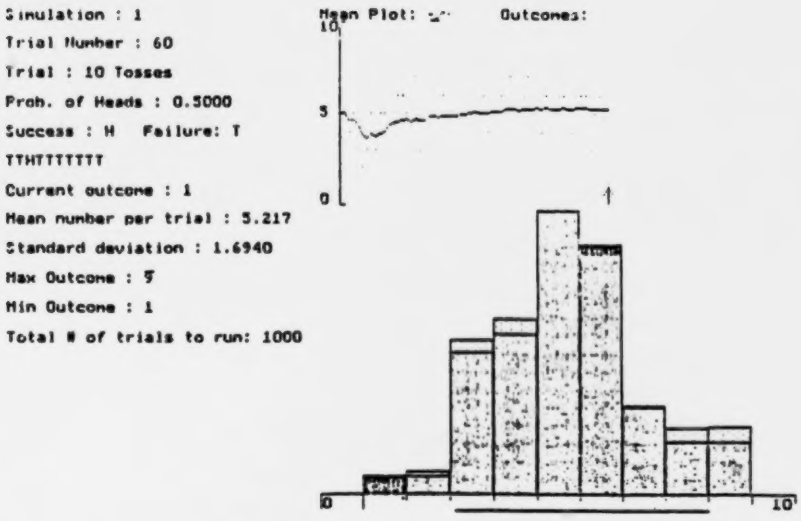
Special Reports

1. *The Mathematical Sciences Curriculum K-12: What is Still Fundamental and What is Not?* (1982). Washington, D.C.: Conference Board of the Mathematical Sciences.
2. *An Agenda for Action: Recommendations for School Mathematics of the 1980's* (1980). Reston, VA: National Council of Teachers of Mathematics.
3. *A Nation at Risk: The Imperative for Educational Reform* (1983). National Commision on Excellence in Education. Washington, D.C.: U.S. Government Printing Office.
4. *Curriculum and Evaluation Standards for School Mathematics* (Working Draft, 1987). Reston, VA:National Council of Teachers of Mathematics.

Articles and Books

5. Schaeffer, R. (1986). "The Quantitative Literacy Project." *Teaching Statistics*, 8 (2).
6. Usiskin, Z. (1985). "We Need Another Revolution in Secondary School Mathematics." *The Secondary School Mathematics Curriculum* (NCTM Yearbook). Reston, VA: National Council of Teachers of Mathematics
7. Gnanadesikan,M., R. Schaeffer, and J. Swift (1986). *The Art and Techniques of Simulation* (the Quantitative Literacy Series - student and teacher editions). Palo Alto, CA: Dale Seymour Publications.

Figure 1



```
Simulation : 1
Trial Number : 60
Trial : 10 Tosses
Prob. of Heads : 0.5000
Success : H   Failure: T
TTHTTTTTTT
Current outcome : 1
Mean number per trial : 5.217
Standard deviation : 1.6940
Max Outcome : 9
Min Outcome : 1
Total # of trials to run: 1000
```

Table 1

```
                                   Frequency Table
Simulation : 1                   Trial : 10 Tosses                   P = 0.5000
Number of Heads      Frequency                Relative Freq          Cumulative Freq
0                    1                        0.0010                 0.0010
1                    11                       0.0110                 0.0120
2                    42                       0.0420                 0.0540
3                    112                      0.1120                 0.1660
4                    220                      0.2200                 0.3860
5                    243                      0.2430                 0.6290
6                    206                      0.2060                 0.8350
7                    115                      0.1150                 0.9500
8                    37                       0.0370                 0.9870
9                    12                       0.0120                 0.9990
10                   1                        0.0010                 1.0000
```

Figure 2

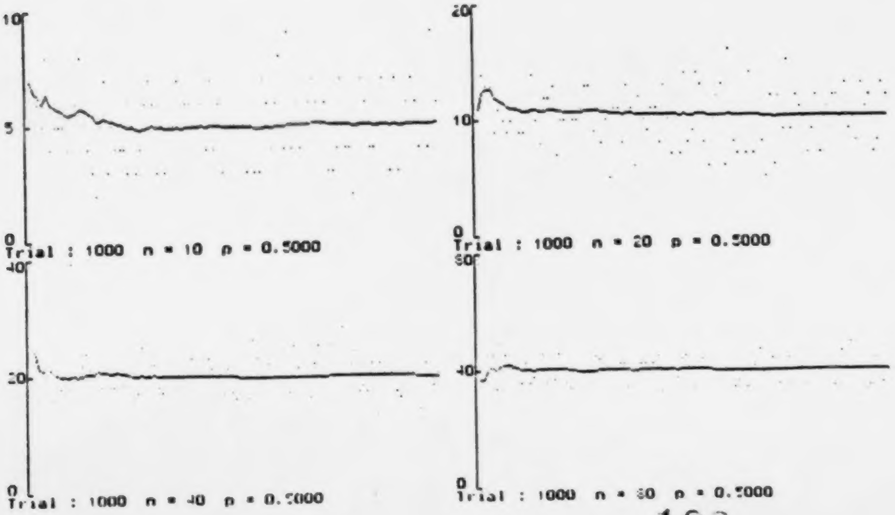


Table 2

Summary Table

#	N	P	Reps	Mean	Std Dev	Max	Min
1	10	0.5000	1000	4.9810	1.55[illegible]	10	0
2	20	0.5000	1000	10.1290	[illegible]	16	[illegible]
3	40	0.5000	1000	20.0450	3.1766	29	11
4	80	0.5000	1000	39.9780	4.4755	54	26

Table 3

Estimator Table

Est1	MAX+MIN-1
Est2	2*MEAN-1
Est3	MAX
Est4	MAX+((MAX-1)/SIZE)
Est5	2*MED-1

Figure 3

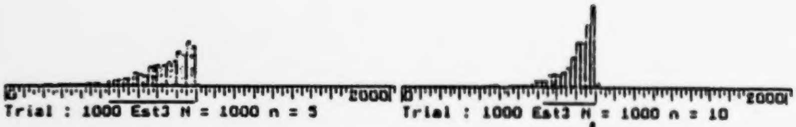
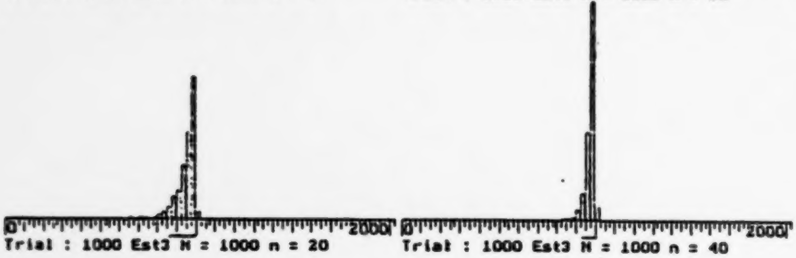



Figure 4

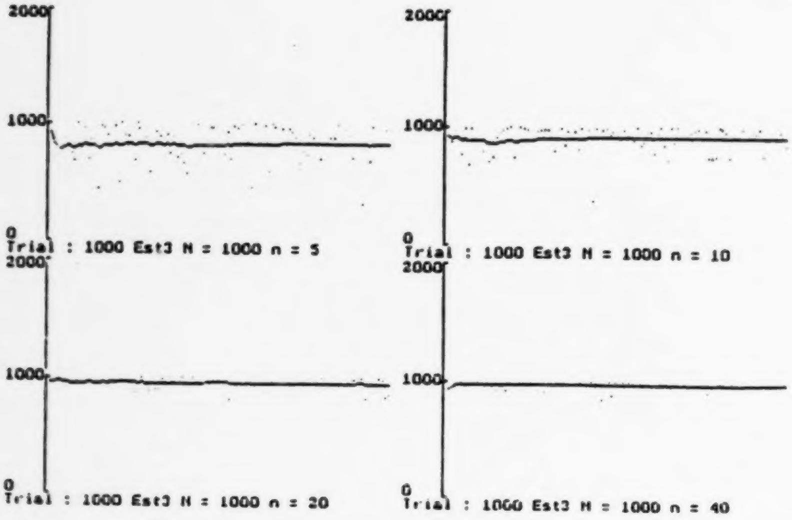


Figure 5

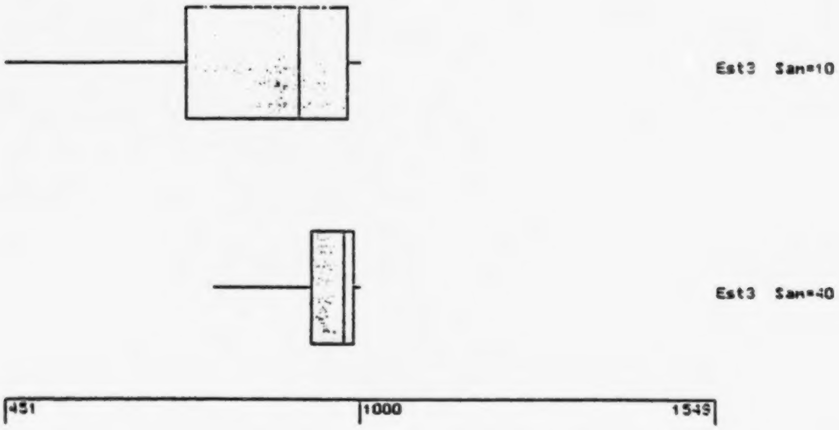


Figure 6

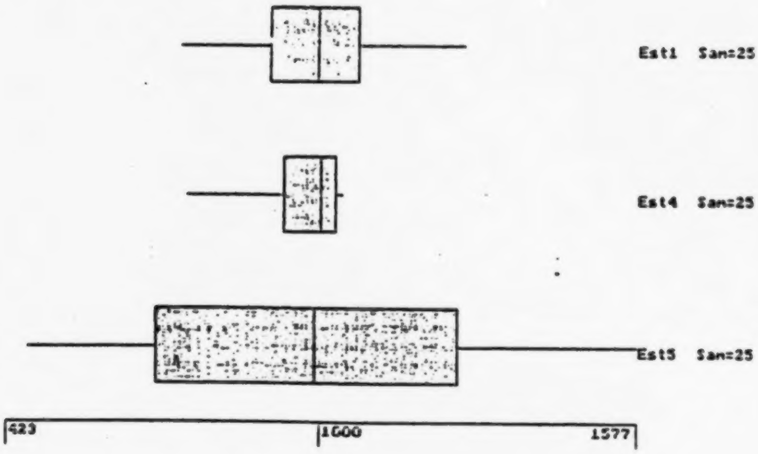
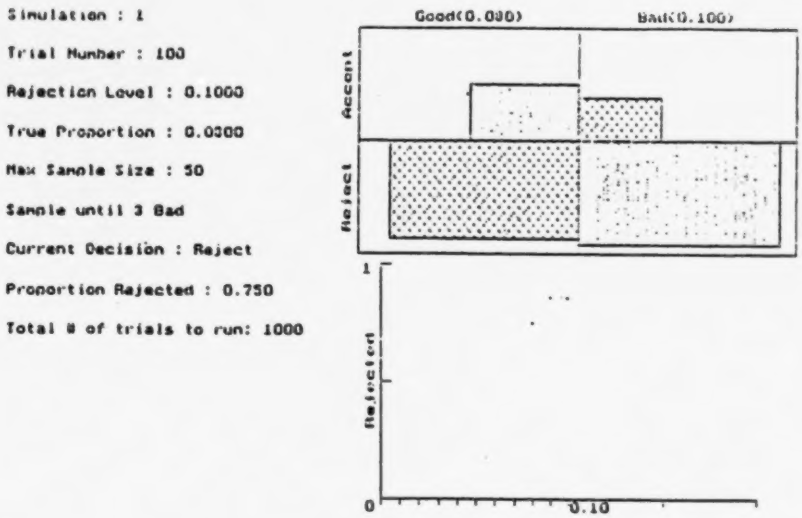


Table 4

Summary Table

Est	Pop	Sample	Reps	Mean	Max	Min	StdDev	MeanEr	MnSqEr
3	1000	40	1000	976.1	1000	849	23.4	23.9	1114.6
3	1000	20	1000	952.9	1000	670	44.8	47.1	4223.3
3	1000	10	1000	907.5	1000	450	67.2	92.5	16148.7
3	1000	5	1000	832.3	1000	224	143.7	167.7	48763.7
3	1000	25	1000	962.2	1000	653	38.5	37.8	2906.4
1	1000	25	1000	997.4	1262	744	54.1	38.9	2929.9
4	1000	25	1000	1000.5	1040	755	37.8	28.3	1430.7
5	1000	25	1000	991.2	1577	465	191.4	156.1	36695.7

Figure 7



```
Simulation : 1

Trial Number : 100

Rejection Level : 0.1000

True Proportion : 0.0300

Max Sample Size : 50

Sample until 3 Bad

Current Decision : Reject

Proportion Rejected : 0.750

Total # of trials to run: 1000
```

Good(0.030)
Bad(0.100)
Accept
Reject
1
Rejected
0
0.10

Figure 8

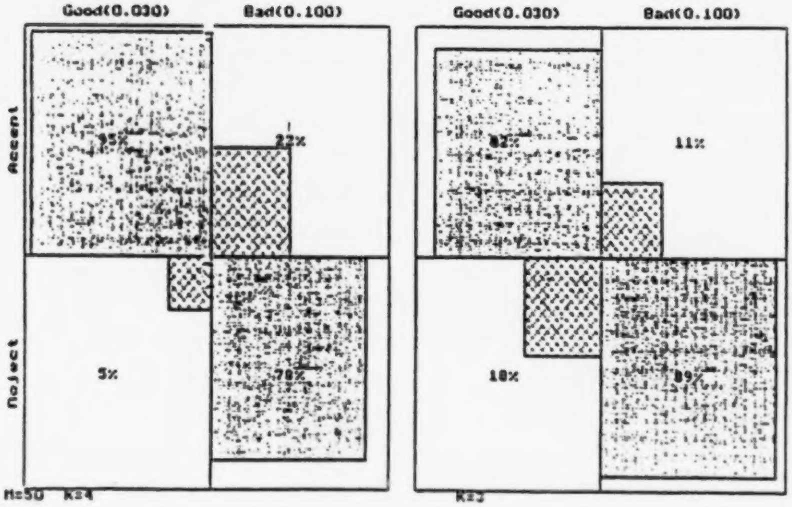
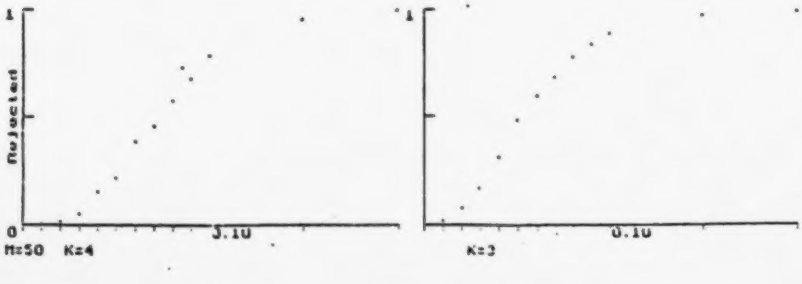


Figure 9



MATH IN OUR LIVES...WITH SPREADSHEETS

Gertrude W. Abramson
Baruch College
City University of New York

Abstract

After more than a decade of educational computing research, agreement is being reached that the teacher is the critical factor in the successful implementation of computer technology in the schools. Now we must identify a viable tool set for teachers to work with and a methodology for using these tools as invisible aids to understanding and productivity in real life situations.

This paper takes an honest look at computing in the classroom and explores the technological options in the learning process. Based upon budgetary, administrative, and inservice training realities, an easy to learn, relatively inexpensive, integrated applications software package is selected as the best of available tools. A method of using the spreadsheet, one of the tools in the integrated package, as a natural component of the learning process follows. The subject is the bank loan process; the precise content can readily be adapted to the age and ability level of the group. Within the lesson, students are given opportunities to manipulate the variables in a loan process, to understand the formulae needed to derive information and to discuss the implications of the various information created through that manipulation. Suggestions are provided for evaluation and for extending the concepts in future lessons.

Introduction

A spreadsheet (SS) is a tool for learning how numbers work together. All too often, in the process of setting up rows and columns and typing data in the appropriate cells, the reasons for using this tool often become obscured. This paper begins with a survey of alternative approaches to educational computing. It proceeds to share a SS-based lesson that allows students to experiment with number relationships important to people living in money-based societies. The lesson has been successfully implemented in classes that bear curriculum labels such as general math, business math and consumer math.

Computing in the classroom

Issues in Educational Computing

At the beginning of this decade, educators were hailing the arrival of the inexpensive microcomputer as the panacea for the crumbling American education system. Today, problems of functional illiteracy and dropout abound; technology has not revolutionized education as it has business. Computers and communications technologies are making teaching and learning more productive in the workplace, in the home and in private settings, but productivity of the schools is declining. Causes for the inherent lackluster of computer- based learning are alternately assigned to teachers, hardware, software, administrators and socio-economic conditions (Lewis, 1988b; Perelman, 1988). Mandinach and Thorpe (1987) place highest priorities on the human factors in technological curriculum development and implementation. Administrative support must include the provision of physical resources and released time for inservice education and for design, development and experimentation with the new media. Peer support, encouragement and participation are required during the design and dissemination phases of the innovative curriculum approaches

Within the academic community, there is increasing conviction that because of the computer's capacity for individualization and interactive participation, computer-based education will be a prominent feature in schools restructured to meet the needs of the information age. As computer educators, we are all aware of the cyclical problems of school budgeting, of the relationships between measured effectiveness and expenditures, of time allocated for teacher-training. In addition, those of us who have been involved in computer education for a decade have witnessed the trial-and-error approach to computing in the school day; contradictory solutions have been proffered for allocation of hardware resources in the schools and for computing as a new discipline versus computing as a tool to improve learning across the curriculum.

Educational Software Options

There are three major software approaches to computer education: educational software (simulations, drill and practice programs and tutorials), programming language tools (BASIC, Logo and Pascal) and productivity tools (word processors, spreadsheets, data bases, communication packages, graphic packages).

A teacher with no computer experience can attend a one day workshop and emerge with sufficient expertise to incorporate educational software, selected by someone else, into the teaching day. However, in this bleakest of alternatives, the school would need a great deal of different software and someone to evaluate existing offerings and make selections. Bork (1985) reported that very little decent computer-based learning material exists and that reluctant computer-using teachers transmit their lack of enthusiasm to their students; just last year, he (Bork, 1987) reiterated this position and added a warning about the widespread acceptance of this poor quality material by teachers and administrators.

Integrated Application Software Packages

Working within the reality of school budgets and teacher preparation time, investment in an integrated productivity tool package will provide the greatest good for the largest number of students. As content-free shells or frames, these tools can be applied in lessons across the grades and curriculum areas. Integrated software programs provide, in one tidy bundle, word processor, database, spreadsheet, graphics and communications; most include on-line spelling checkers.

PFS First Choice (Professional Series, 1986), for use on the IBM-PC and compatibles, is the package used in this lesson. I particularly like the minimal learning time needed, the simplicity of the screen vocabulary prompts, the on-line help facilities that really help and the relatively low price. Software reviewers have echoed my findings and conviction that if a package is good for a beginner working at home, it is appropriate for classroom use also. Zachmann (1987) states that PFS First Choice meets the needs of the beginning user interested in a variety of applications in an inexpensive package on a microcomputer. Palmer (1987) adds that it sets a new standard of user-friendliness, automatically adjusting column widths, allowing easy formula entry and "styling" numbers as dollar amounts, number of decimal places, commas, and percent displays and Bryan (1988) notes that nearly all program functions are accessible from concise menus.

Davidson (1988), a teacher who is now an educational software publisher, attributes her success with computers in the classroom to her early recognition of the technology as a tool to learning. Snyder (1988) cites the computer's incredible impact on business through the use of the very productivity tools available for use in the classroom. Lewis (1988a) describes a "model school" in which computers have been integrated into every aspect of the curriculum, especially in the seventh and eighth grades. The main software in use at that school is Apple Works, a program that combines a word processor, a spreadsheet and a data base; the computing facility consists of 30 Apple IIGs. The lesson described below could be implemented with Appleworks with only cosmetic modifications.

Spreadsheets and bank loans

Students who have been assigned to vocational or general high school programs frequently suffer from poor self-images. They need infusions of positive experiences that build confidence in their abilities to succeed in the adult world. These experiences can readily be developed with the use of software applications tools as invisible aids in the learning process. Used appropriately, these tools can assist in the understanding of real life situations and help in the process of nurturing competent adult citizens.

Following is a very detailed lesson plan for teaching about loan procedures with spreadsheets as a tool in the process. The BARUCH model (Abramson, 1987) is used to organize the components of lesson development.

Begin With a Goal

Students will be given opportunities to manipulate the variables in a loan process, to understand the formulae needed to derive information, and to discuss the implications of the information created through that manipulation.

Analyze Learner Situation

Learners are boys and girls, uninspired 9th and 10th grade general math students. Strengths students bring to this situation are experience with computer keyboarding, previous SS use in this environment, and positive attitudes toward computer-based learning activities. Weaknesses to be overcome include discomfort analyzing and solving story problems, difficulty dealing with abstractions, and aversion to anything called "math".

Redefine Goal as Behavioral Objectives

1. Students will define the vocabulary of bank loans.
2. Students will compose a list of "what-if" scenarios.
3. Students will set up spreadsheet as per instructions.
4. Students will type labels, data and formulae.
5. Students will restate each formula as a question.
6. Students will discuss the implications of borrowing moneyat different rates and terms.

Use Instructional Materials

1. IBM-PC or compatible computers, double disk drives andprinter access, 256K memory.
2. PFS-First Choice software.
3. Student file disks.
4. Whiteboard with markers and marker-eraser.
5. Vocabulary sheets; worksheets.

Preliminary activities must provide students with the background needed for a meaningful experience and create a positive climate for learning. An excellent example was supplied in 1987 by General Motors Acceptance Corporation (GMAC) who offered four year car loans at an annual rate of 2.75% when bank car loans were going at about 10%. Invite students to share experiences they or people they know have had with trying to get a bank loan, trying to maintain payments, etc. Present the problem statement and encourage discussion:

> Next month, for my seventeenth birthday, I am going to get a set of wheels. I have been working at Shop Rite since I got working papers at fourteen and everyone knows I am very reliable. I went to three banks with a letter from my supervisor and in each bank I was told I can get a car loan if someone cosigns for me. My uncle says if I can prove to him that I can make the payments, he will cosign for me. How can I figure out now how much money I have to pay back each month? The more money I

can borrow, the more I can spend on a car. How can I find out how much I can borrow?

Before proceeding to the computer, discuss each item on the vocabulary list (Figure One), Make sure students can distinguish data (numbers needed as input) from information (numbers to be derived by process).

Computer activities must accommodate the number of workstations available for class use. This lesson was designed for two students per workstation but has been used in a cooperative environment with four per station. One teacher conducted the lesson with a single classroom computer; although I though the lesson was "watered-down" in impact, the classroom teacher disagreed. Housekeeping tasks of distributing disk sets load the program should be done quickly and quietly in a routine manner.

While you create columns on the whiteboard, have the students label columns in the heading area, the first three for data and the last three for information. Use abbreviations to keep the columns from getting too wide: principal (PRIN), annual interest rate (RATE), term of loan in months (TERM), monthly payment (MPMT), total debt (TOTDEBT) and cost of borrowing the money (COST).

Present a few "what-if" situations:

What if I borrow $1000 at 10% over 12 months?
What if I borrow $1000 at 10% over 24 months?
What if I borrow $1000 at 5% over 12 months?
What if I borrow $1000 at 5% over 24 months?

Let the students dictate the whiteboard columns in which you will enter the data while each workgroup enters the data appropriately in the SSs. Style each column (Figure Two).

Explain that a formula is a process that takes data and creates information. Every formula can be restated as an English language question. Formulae with difficult computations are built into SS programs; simple formulae are often supplied by the user. Write the payment formula and explain the words and ideas involved. The students need to know *what* a formula does and *why* and *when* it is used; *how* that formula works requires a higher level of abstraction and mathematical sophistication. Encourage the students to help you derive the formula for total debt and total interest charge; restate the formula as questions:

PAYMENT ON PRIN AT RATE/12 OVER TERM (Formula for MPMT cell): What is the monthly payment on the amount borrowed at the annual interest rate over the number of monthly installments?

MPMT * TERM (Formula for TOTDEBT cell): How much money will I end up giving to the bank? What is the amount of the total debt I have agreed to pay to the bank?

TOTDEBT - PRIN (Formula for COST cell): How much will it cost to borrow this amount of money from the bank?

After the formulae are typed into the appropriate cells and the information appears on the spreadsheet screen, encourage the students to compare results, to add a few more "what-if" situations and to save and print the SSs (Figure Three). Use the derived information for a discussion:

1. Why does a bank lend money to people?
2. How does a bank make its money?
3. What happens to the amount of each payment as the number of payments become greater?
4. When is better for the borrower to stretch out the payments as long as possible?
5. Why might someone prefer a short term loan?

Complete Lesson with Evaluation

Mastery of this lesson may be measured with other "what-if" questions using the SS created during the lesson. In addition, a paper/pencil or oral examination might address the following issues: list the variables needed to compute a bank loan, explain the processes used to determine desired information, explain the ramifications of borrowing money at different rates and for various periods of time.

Highlight Presentation with Enthusiasm

Good lessons require careful preparation; a "dry-run" with colleagues during prep time or with classmates in a graduate or inservice course would be ideal. Every teacher who has used some form of this lesson has discovered a list of similar activities that would be appropriate for his/her class. Enthusiasm, in general, is contagious and enthusiasm for computer-based learning is conveyed at electronic speed. Show your students that you are giving them a tool whose use will continue far beyond the walls of the classroom. You are then on your way to a great learning experience.

Conclusion

This lesson generated a great deal of excitement. With young students, the problem situation read, "My mom says we can buy a car...She has been working for the gas company for five years...etc." In an adult education class, the car loan was changed to a home improvement loan based upon the interest of the students.

SSs in the math curriculum afford math-phobic students and teachers operation-free opportunities for manipulating numbers and for gaining insights into the processes involved. This is a tool for learning to solve problems, for improving organization and thinking skills and for gaining experience with careful expression. An extension lesson might set up an amortization table and discuss the relation of interest to debt reduction as the debt draws closer to completion. With this lesson, we have transmitted to our students a tool for understanding the process of borrowing money; we have given them the building blocks needed to cope with the pitfalls of dealing with loansharks and the implications of overextended credit card use.

References

Abramson, G. (1987). Planning for computer enhanced instruction. PA: *Proceedings of the National Educational Computing Conference*, pp. 273-276.

Bork, A. (1987). *Learning with Personal Computers*. NY: Harper and Row.

Bork, A. (1985). *Personal Computers for Education*. NY:Harper and Row.

Bryan, M. (1988). Software that does it all. *Personal Computing*. April 1988, pp. 95-98.

Davidson, J. (1988). *Computer Classroom*. CA: Davidson and Associates, *3* (1).

Lewis, P. (1988,a). 'Model' for U.S. schools. *The New York Times*. September 6, C12.

Lewis, P. (1988,b). The computer revolution revised. *The New York Times* August 7, 4A, 15-17.

Mandinach, E. and Thorpe, M. (1987). Caveats and realities in technological curriculum innovation. *Technology and Learning, 1* (4).

Palmer, S. (1987). The "lightweights" strike back. *Business Software Review*. August 1987, pp. 24-25.

Perelman, L. J. (1988). *Technology and Transformation of Education*. VA: National School Board Association (NSBA) Institute for the Transfer of Technology to Education.

Professional Series. (1986). *(PFS) First Choice* for the IBM Personal Computer (PC) and compatibles. CA: Software Publishing Corporation.

Snyder, T. (Winter 1988). *The Professional Teacher: The Newsletter for K-12 Educators*. MA: Tom Snyder Productions.

Zachmann. W. (1986). 'pfs: First Choice' for novice business users. *Software News*, October 1986, pp. 32-33.

Figure One: Bank loan vocabulary

1. The *bank* is the lender of the money and the place to which you send your payments.
2. The *borrower* is the person who gets the loan and must repay it with interest in monthly payments.
3. The *cosigner* agrees to be responsible for everything you owe the bank if you default (cop out).
4. The *principal* is the amount of money you are borrowing, the loan.
5. The *annual interest rate* is the yearly charge for borrowing the principal. To get the monthly interest rate, we will divide interest by twelve (interest/12).
6. The *term* is the number of months it will take to pay off the debt.
7. The *debt* is the principal and interest together; the total amount you must pay to the bank.
8. The *monthly payment* is the debt divided by the term.
9. The *cost* of borrowing the money is the difference between the total debt and the principal, the total amount of interest paid.
10. To *amortize* is to liquidate (get rid of) a debt by installment payments.

Figure Two: Representing numbers

For money numbers:
Currency symbol: $
Show currency symbol - Y
Show commas - Y
Show % - N
Number of decimal places - 2.

For interest rate:
Currency symbol: $
Show currency symbol - N
Show commas - N
Show % - Y
Number of decimal places - 2.

For term in months:
Currency symbol: $
Show currency symbol - N
Show commas - Y
Show % - N
Number of decimal places - 0.

Figure Three: Bank loan spreadsheet

```
    PRIN        RATE    TERM    MPMT      TOTDEBT      COST
$1,000.00     10.00%     12    $87.92    $1,054.99    $54.99
$1,000.00     10.00%     24    $46.14    $1,107.48   $107.48
$1,000.00      5.00%     12    $85.61    $1,027.29    $27.29
$1,000.00      5.00%     24    $43.87    $1,052.91    $52.91
$5,000.00     10.00%     48
$5,000.00      2.75%     48
```

STATE OF THE ART
AN EDUCATIONAL SOFTWARE DESIGNERS' FORUM

Dave Grady, *Chair*
Editor
The Grady Report
Redwood City, CA

ABSTRACT

The Software Designers' Forum has been on the drawing board for the past two years. Discussions have focused on the need to evaluate past efforts, discuss design issues that effect current development, and make some projections about future challenges. Unlike a panel format with formal presentations, the Forum will encourage informal discussion among participants in response to questions presented by the Moderator. A question will initially be addressed to two Forum participants and after their brief responses, will be opened up to the entire Forum.

The discussion questions will cover the following points:

- Trends Toward the Standardization of Interface
- The Research/Design Connection
- Effects of New Technologies
- Hypermedia Design Issues
- Designing Products for the 1990s

FORUM PARTICIPANTS

Walter Koetke
Director, Research and Development
Scholastic
New York, NY

Lynne Anderson-Inman
Division of Teacher Education
University of Oregon
Eugene, Oregon

Midian Kurland
Director of Software Development
Bank Street Software
Cambridge, MA

Mary Cron
Vice President, Product Development
First Byte
Long Beach, CA

Mitch Resnick
Designer/Talking Software
Doctoral Candidate/ MIT Learning Group
Cambridge, MA

Susan Schilling
Development and Creative Director
MECC
St. Paul, MN

Tim Barclay
Curriculum Development
TERC
Cambridge, MA

Michael Hillinger
Executive Editor
Houghton Mifflin
Hanover, NH

SPONSOR: ICCE

DISTRICT-WIDE COMPUTER IMPLEMENTATION

Marilyn Gardner
Sheryl Asen
Hilary Cowan, Judy Clifford, Linda Lindroth

ABSTRACT: P.S. 2000: An Action Plan for Using Technology to Educate Students for the 21st Century

Marilyn Gardner, Boston Public Schools, The Humphrey Center, 75 New Dudley Street, Boston, MA 02119.

Boston Public Schools response to addressing the technological needs of education in Boston Schools is a project called P.S. 2000. P.S. 2000 is a twelve year vision which encompasses thirteen major projects and empowers teachers, parents, and schools through a new school environment which encourages teachers to embrace technology to improve instruction and productivity.

P.S. 2000 sets a new vision in which learning for subject mastery is accomplished through and with computers and related technologies. Teachers and parents have opportunities for greater access to information and more varied learning resources to improve the way students learn.

The mission of P.S. 2000 is to develop a clear, systematic, progressive, and comprehensive model for the instructional and administrative uses of technology in Boston Schools. P.S. 2000's thirteen major projects, which represent a powerful array of technological strategies, will help Boston schools reach three fundamental goals: improve the quality of learning; increase equity of educational opportunity and access; and ensure greater cost effectiveness.

P.S. 2000 is especially unique in that the Boston Business Community has joined efforts with Boston Public Schools to make P.S. 2000 a reality. The Boston Business Community realizes from personal experience the extensive costs associated with technologically upgrading their business and offices, and that funding for instructional and administrative technology tools and equipment is out of reach for most public education budgets. The Boston Business Community has joined that leadership of the Boston Public Schools to provide Boston studnts with the additional intellectual tools they will need to move into the 21st century.

ABSTRACT: The Computer Integration Project in the Arlington, VA Public Schools

Sheryl Asen, Specialist, Academic Computing, Arlington Public Schools, 1426 N. Quincy Street, Arlington, VA 22207.

The *Computer Integration Project* was developed to help meet the goal of integrating computer use into instruction. Academic Computing specialists work with curriculum supervisors, principals, and teachers (all of whom may submit a proposal) to ensure that specific software packages are used as instructional materials to help students master the instructional goals and objectives of specific courses or units of study.

Academic Computing provides support through the proposal, training, implementation, and evaluation processes and funds the projects. Proposals must be endorsed by all participants and must clearly state: specific computer software, peripheral equipment, and ancillary materials needs; specific relevant curriculum objectives; how the software and materials will be used in instruction (and with the number of computers currently available); the type and duration of training for participating teachers. Projects are evaluated for consideration for revision or expansion.

Strengths: match between software and instructional objectives is explicit; most software used promotes higher level thinking and user control; staff involvement in planning and evaluation is increased; teachers participate in additional projects.

Weaknesses: curriculum staff involvement in projects initiated by schools is minimal; successful projects are not being expanded to other sites; some resistance by administrators to non-traditional teacher inservice training.

ABSTRACT: Implementation of Computers in Classrooms: The Plan and the Progress

Hilary Cowan, Judy Clifford, Linda Lindroth, Kanawha County Schools, 200 Elizabeth Street, Charleston, WV 25311.

Marked inequities in computer equipment, use, and teacher commitment in Kanawha County's 76 elementary schools spurred the creation of a four-year plan of computer implementation. By 1981, approximately 1400 elementary teachers will have received computer hardware and software for their classroom along with concomitant training. The plan, which costs approximately $300,000 annually, also provides stipends for two essential personnel components: a team of 10 Teacher Technologists who train teachers, and 76 Computer Specialists who coach and support those receiving training in their respective schools.

In order to provide teachers with release-time training, the Charleston Regional Chamber of Commerce and Development has joined forces with Kanawha County Schools to create Project TEACH, a unique project designed to bring business people into the schools while teachers leave for computer training. Armed with their own 3-hour teaching plans, over 450 employees from 90 businesses have substituted for the teachers. The slogan "Education Means Business" has been exemplified in this added dimension to the four-year plan.

The enthusiasm of both teachers and students is building as the plan progresses. In a county where funds are at a premium and where many small schools are necessary to serve outlying areas, structured planning has been the key to counteracting these disadvantages and maximizing the success of equitable computer implementation.

AWARD-WINNING EDUCATIONAL SOFTWARE AND CURRICULUM INNOVATIONS IN HIGHER EDUCATION

Robert Kozma
Jerome Johnston
NCRIPTAL
The University of Michigan
Ann Arbor, MI

ABSTRACT

EDUCOM and NCRIPTAL (The National Center for Research to Improve Postsecondary Teaching and Learning at The University of Michigan) conduct an annual competition in which they award $5,000 to the best software and computer-based curriculum innovations. Judgments are made by a national panel of experts in instructional design and in the academic disciplines. This presentation describes the awards program and the dimensions of educationally sound software and classroom implementations. It includes video vignettes of winning entries.

SPONSOR: EDUCOM

CASE STUDIES OF COMPUTER USE: SPECIAL EDUCATION STUDENTS IN MAINSTREAM SETTINGS

Judith Zorfass
Education Development Center, Inc.
55 Chapel Street
Newton, MA 02160

ABSTRACT

This presentation is based on over two years of naturalistic research into the technology integration process in four diverse middle schools in eastern Massachusetts. Funded by the U.S. Department of Education, Office of Special Education Programs, the Education Development Center (EDC) and Technical Education Research Centers (TERC) are engaged in a three-year investigation (October, 1986—October, 1989) of computer use with mildly handicapped students in mainstream and special education settings. The goal of the project is to develop recommendations so that administrators, specialists, and teachers can consistently integrate computers into the curriculum in such a way as to meet the needs of mildly-handicapped students.

Having closely followed mainstream teachers as they integrated computers into the curriculum, EDC/TERC has written a set of case studies depicting situations where teachers have either successfully or unsuccessfully met the needs of handicapped students. From our analysis of these positive and negative cases across sites, several factors related to teacher practice have emerged. This presentation will focus on three related factors that we believe can contribute to successful integration: (1) a teacher's ability to recognize and understand handicapped students' strengths and weaknesses in developing and applying skills and processes; (2) the teacher's ability to use computer-based activities as a context for ongoing assessment of students' strengths and weaknesses; and (3) a teacher's ability to adapt and modify computer activities based on this diagnostic information about learner characteristics.

The presenter will discuss these three related factors in depth by contrasting positive and negative examples drawn from our case studies. For example, we will describe one social studies teacher who, over the course of a year, successfully modified her use of *Where in the World is Carmen San Diego* based on her growing awareness of students' difficulties. In contrast, we will describe an English teacher who provided a learning disabled student with a word processor for all written assignments, yet the teacher did not fully understand the extent of the student's difficulties with written expression, and did not adapt computer use in ways that would meet the student needs.

The audience will be invited to discuss each illustrative case and its implications for successfully integrating technology into instruction.

TECHNOLOGY AND MANIPULATIVES FOR EARLY CHILDHOOD AND SPECIAL NEEDS STUDENTS

D. Midian Kurland, Marilyn Quinsaat
Nancy Corning, Jack Halapin

ABSTRACT: Early Childhood Initiative

Marilyn Quinsaat, D. Midian Kurland, Bank Street Software Group, 55 Wheeler Street, Cambridge, MA 02138.

The Early Childhood Initiative (ECI), conducted by Bank Street College of Education, is a formative research and development project studying issues related to the use of multi-media technologies in pre-school settings. The ECI is focusing on three concerns regarding the potential for a multi-media environment in pre-schools:

1. To what extent can different media broaden the perception of the young child's world?
2. To what extent is the pre-school environment altered or a new environment created for children?
3. In what ways can the media empower young children to do tasks that they are unable to do without the technology?

The ECI is supported by equipment donations from Apple Computer, Inc. The presenters will describe efforts to implement software and hardware systems in a Cambridge pre-school and will address additional issues such as: What interface conventions are appropriate for this age group? In what way does the environment provide for more collaboration? What is the impact of art-tools as a precursor to emerging literacy skills? Finally, the presenters will recommend design elements of systems which address successful implementation and the needs of the young child.

ABSTRACT: Bridging the Learning Gap—Early Primary Computers and Manipulatives

Nancy Corning, Jack Halapin, Ridgefield Public Schools, Box 629, Ridgefield, CT 06877.

An appropriate Early Childhood curriculum must effectively "bridge the learning gap" which currently exists between the young child's concrete world and abstract academic requirements. This presentation will show how the computer, when used with appropriate manipulative material and carefully selected software, can be a powerful ally in assuring the ease of this important transition. Because the computer itself has both concrete and abstract qualities, it can provide the perfect link to align tangible, concrete manipulative activities with abstract curriculum demands.

Also included in this presentation will be a variety of original techniques which we have devised and found very successful when dealing with computers and the very young (often non-reading) student. These unique strategies will allow a teacher, using one computer and no additional staff, to effectively integrate the computer into the routine of a standard primary classroom.

This concept coordinates and unites two highly effective teaching tools, computers and manipulatives, for the benefit of the child, who not only feels more comfortable, confident, and in control of learning, (no more "math-phobics" and "computer-phobics"), but is more likely to retain and transfer information, since the young child truly learns best by "doing".

COMPUTER NETWORKS FOR LEARNING

Reed A. Augliere
Katherine Flannery Moss
James R. Samson

ABSTRACT: How Computer Networks Transform Learning Environments

Reed A. Augliere, Harvard University Graduate School of Education
and Harvard University Office for Information Technology, Room 519 Gund Hall, Cambridge, MA 02138.

While researchers have shown considerable interest in the effects of computerization on education, a clear model of how computers transform the communications structures of school environments has yet to emerge. In particular need of clarification is the relationship between such structures and learning behaviors. As standalone microcomputer systems become increasingly linked into multi-system networks, this deficiency will become more and more deeply felt.

Drawing upon exchange and communications theory, diffusion of innovation studies, and social network analysis methodology, we have synthesized a conceptual framework that represents a first step toward a remedy. In this approach the units of analysis are not individuals but communicative ties, defined as exchange relations; the object of exchange is information, which functions to reduce the situational uncertainty of social actors.

From diffusion studies come the notions of innovation (computer networks) as introducing uncertainty into social contexts and contagion as the process by which innovation-related information moves through communicative structures. The probable transformation of these structures, and subsequent impacts on learning outcomes, can then be modeled under a variety of behavioral assumptions.

We will demonstrate the utility of such an approach in our discussion of a project in progress.

ABSTRACT: Project SHINE

Katherine Flannery Moss, Boston Public Schools, Technology Support Specialist, 75 New Dudley Street, Boston, MA 02119.

Project SHINE, funded by the Massachusetts Department of Education, used telecommunications to break down the physical barriers to communications between seven communities. The development of a regional newspaper allowed teachers and students to focus on regional concerns, and also explore the promise of this new technology.

During the 1987-1988 school year, students and teachers in grades six through twelve from seven Boston area cities and towns participated in Project SHINE. Participants in SHINE developed environmental, nuclear awareness, and multicultural units to share between schools using telecommunications. The goals of the project were to establish the networks necessary for student sharing on environmental issues and to publish a regional newspaper of student writing on these issues.

After six hours of telecommunication as training, teachers used their skills to introduce themselves on a public message board, The Physics Forum. The Forum, as well as DELHI Boston, WGBH Online, and Applelink, were the electronic Bulletin Boards that were used in the project. Students exchanged ideas and data on topics such as water pollution, Boston Harbor sealife, immigration surveys, human mate selection survey, and folktale exchange. The highlight of the project was an interactive teleconference on water pollution in which all seven committees participated. Students' reaction to this was extremely positive and enthusiastic.

The presentation will include a discussion of the major issues involved in developing the project including:

- Telecommunications and teleconference logistics;
- Overview of bulletin boards used in the project;
- Strategies developed for working with several school systems;
- Teacher training model.

ABSTRACT: Kids' Network

James R. Samson

In a Boston school, 4th and 5th graders became geographers, researchers, statisticians, and environmentalists, using the National Geographic's Kids Network. The Network is a nationwide system that gives students a chance to communicate using computers. The children gathered data about "pets and games", sharing and comparing their data with schools throughout the U.S. and Canada, through the use of a modem. As information was obtained the children would plot their data on computer maps and graphs. The maps allowed the children to pinpoint "global addresses" through longitude and latitude read-outs. A modified AppleWorks was used both for the telecommunication of information and letter writing.

The skills developed during the 'HELLO' project prepared the children for the 'ACID RAIN' project. During this unit the children utilized their computer process, experimented with pH scales, and then built their own rain collectors to gather data on rainfall in Boston. As other children from throughout the country also gathered data, the children corresponded with one another through the telecommunication link. Information was plotted, surveys taken, and many insights were developed regarding the problems of acid rain throughout the world. Children also gathered information from other parts of the world as well as being able to communicate with Unit scientists when needed. Students used technology to learn valuable scientific and environmental lessons.

FACILITATING TECHNOLOGY-RELATED RESEARCH IN A SCHOOL SETTING

Vivian Johnson
MECC
University of Minnesota
Center for the Study of Educational Technology
3490 Lexington Avenue North
St. Paul, MN 55126

Greg Sales
University of Minnesota

Randall E. Johnson
Anoka-Hennepin Ind. School District #11
Coon Rapids, MN

ABSTRACT

This panel session will stress the need for cooperation among various members of the educational community to address the major problems inherent in conducting technology-related research.

One basic problem is that technology has been used in schools for many years but little empirically based research judging its effectiveness has accumulated. Dr. Peggy Roblyer of Florida A & M University states: "The overwhelming conclusion that you must come to is that there is not enough [technology-related] research to make good conclusions" (*World Leaders*, 1988, p. 4). This lack of empirical evidence contributes to the growing apprehension about the value of continuing or expanding support for technology-related programs.

Complicating this is the fact that technology-related research has been hampered by methodological shortcomings noted by researchers such as Dr. Henry Becker of Johns Hopkins University (Becker, 1987). Shortcomings include selecting student samples for the convenience of school class groupings instead of through a randomizing method, small sample sizes of typically fifty or fewer students, treatments of only a few weeks, and the lack of comparison groups to contract to the technology-using students.

Finally, researchers and school personnel must work together to find creative solutions to the lack of school participation in technology-related research projects. Facilitating a dialog among different members of the educational community would include the following issues:

- additional time and effort required by teachers participating in research projects;
- scheduling the experimental treatment to coincide with an appropriate period in the school's curriculum;
- limiting unexpected school scheduling changes that can disrupt research plans.

Building a solid technology-related research base requires cooperative efforts to remove the traditional difficulties associated with field-based research.

Panel members will address three points:

- identifying problems associated with conducting school-based research from the perspective of public school personnel and university researchers;
- suggesting possible solutions to these identified problems;
- developing cooperative school, university, and corporate partnerships to facilitate research.

CRITICAL THINKING

Esther Inbal, Oded Susskind
Peter Skillen
Kim MacGregor, Judi Repman

ABSTRACT: Categorization and Data Modeling with Classification Machine

Esther Inbal, Oded Susskind, Boston University.

This presentation demonstrates a learning environment focusing on the use of microcomputers for teaching the concept of classification in the elementary and middle school science curriculum. The Classification Machine is a shell program which can be used by student to classify and categorize objects in any subject or field. It is an interactive software for designing and creating new classification key as well as for defining unknown items with existing keys.

We discuss the approach of using the computer as tutee. The student "teaches" the computer and by doing it acquires thorough knowledge of the subject and ideas about the process of learning. The computer-based system "mirrors" the students' thinking processes, helping them to reflect on those processes and thereby to improve their metacognitive skills.

The Classification Machine utilizes the computer ability to record and display the classification process in a visual interface. The graphical presentation of the decision tree allows the user to perceive the organization of the guide and modify the tree while it is being created. The program emphasizes the designing process rather than the end product.

Classification Machine is aimed at third to tenth grades. It is also of value in teacher training programs at the college level.

ABSTRACT: THINKINGLAND: An Intentional Learning Project—The Database and the Culture

Peter Skillen, Program Leader, Computers in Education, 550 Finch Ave. W., North York, Ontario, Canada M2R 1N6.

The ThinkingLand project focuses on the development of an environment that will promote Intentional Learning (Scardamalia & Bereiter, 1983) among students. Intentional learners take an active role in pursuing knowledge about, and in regulating, their own learning and thinking strategies. The *Thinkingland Database*, written in HyperCard, is central to this environment and serves two major purposes. The database fields (or prompts) provide executive scaffolding for the novice by giving information as to what an expert might think about as they perform a task. Students can think about and record their own thoughts about their plans, bugs, questions, triumphs, and reflections regarding a task. In addition, the database provides students with the capability to look at other students' information. Here is a situation, therefore, where students not only get a chance to see what other peoples' final products look like, but they get an opportunity to '*watch*' the process that other students go through in performing a task. The initial plans are available to see. The changes in the plans are made explicit, as are the reasons why.

This is currently being developed for use with students who are learning the Logo language (LogoWriter) at the Junior/Intermediate grades. It is the intention that a general cross-curricular tool will evolve from this research and development.

It is imperative that we find ways of allowing students to 'take charge of their own minds'. ThinkingLand is one such attempt at using the technology to achieve that goal.

ABSTRACT: Developing Critical Thinking Skills: A Computer-Assisted Intervention Program for At-Risk Students

Kim MacGregor, Judi Repman, Louisiana State University, 111 Peabody Hall, Baton Rouge, LA 70803.

Dropping out of school, a national concern, is a critical problem in the state of Louisiana. CATS (Computer-Assisted Thinking Skills) is a dropout intervention project being implemented this year in one school system in the state. The focus of the program is on providing middle school students computer-assisted instruction that facilitates the development of critical thinking skills in the math, science, and social studies curricula. Critical thinking skills in each of these content areas were identified, and instructional software was selected on the basis that it could be used to teach the targeted thinking skills and was appropriate for existing curriculum topics. The computer activities were designed to be an integral part of the instructional program and framing and bridging activities were included. The project was implemented in four phases including teacher in-service, curriculum development, implementation with students, and evaluation. A description of the teacher training modules and the curriculum development process, as well as examples of learning activities in each of the content areas, will be presented. In addition, a summary of student outcomes including content area achievement, critical thinking, self-esteem, attitude toward school, and daily attendance will be provided.

RESEARCH THEMES IN ARTIFICIAL INTELLIGENCE AND EDUCATION

Chris Dede
University of Houston--Clear Lake
2700 Bay Area Blvd.
Houston, TX 77058

Andy Molnar, Program Director
Applications of Advanced Technology
Science and Engineering Education
National Science Foundation

Joe Psotka, Chief
Smart Technology for Training
U.S. Army Research Institute

ABSTRACT

As the power of computer hardware increases, growing numbers of educational institutions can afford devices capable of supporting artificial intelligence (AI) applications. Simultaneously, advances in cognitive science are enabling the development of software with embedded, sophisticated teaching/learning models. As these applications are deployed, the research agenda in educational computing will alter to emphasize the challenges and opportunities posed by this shift from conventional computer-assisted instruction.

One system for grouping these emerging approaches matches each research area with a long-standing instructional philosophy:

- self-directed exploration: microworlds
- easy access to knowledge: hypermedia
- facile accomplishment: empowering environments
- individualized instruction: intelligent tutoring systems
- peer tutoring: computer-supported cooperative learning
- bridging geographic barriers: distance learning

Much of current research in educational AI is directed to actualizing these historic instructional goals.

This session will focus on evolving research themes in AI and education. Three different perspectives will be presented, each reflecting:

a. the historical development of instructional AI;
b. current state-of-the-art projects;
c. probable future research directions; and
d. gaps and needs in existing work.

This discussion will be targeted to how AI applications in education are progressing and how they may impact the overall direction of instructional computing research.

TOUCH TYPING IN ELEMENTARY SCHOOLS—WHY BOTHER?

Pamela Freyd
Coordinator, Computer Education

Jessica Kahn
Lecturer

Graduate School of Education
University of Pennsylvania
3700 Walnut Street
Philadelphia, PA 19104
(215) 898-5690

Abstract

Teachers are finding word processing to be a valuable tool in writing instruction. Current debate focuses on the merits of introducing formal keyboarding skills in the early grades (Balathy, 1988; Warwood, 1985; Suttles, 1983). This paper examines the issue of keyboarding in elementary schools.in terms of the purposes of computer use as well as the effectiveness of keyboarding instruction.

Drawing from our research, we examine the arguments for such instruction in terms of children's abilities (developmental issues) as well as available resources (ecological considerations). We question whether the investment of time and effort is a worthwhile use of limited resources. A formidable investment of resources is required for children to master the techniques of touch typing: whole class instruction (one child per computer) including several thirty minute sessions a week over at least a six week period (Warwood, 1985; Milkes, 1985). We do not question that, with such an investment of time and equipment, children can be trained to be touch typists. We do, however, question whether touch typing is a necessary prerequisite skill for writing with computers in elementary schools.

Our paper, based on three years of research in two public elementary schools in which computers were used for writing, refutes the claim that keyboarding instruction is necessary for elementary school children. Even with time-consuming instruction concerning the correct fingering, children's typing speeds rarely surpass their handwriting speeds. Our research (with children K-8) indicates that with brief introductions to the locations of the keys and hunt-and-peck strategies children type more quickly than they can write by hand. This ability to produce text easily allows children to concentrate on composition concerns.

INTRODUCTION

Teachers are finding that word processing is a valuable tool for writing instruction. This recent awareness is probably related to the increasing number of computers in schools. Just three years ago, word processing and writing accounted for 5% of computer use in one survey of schools. (Becker, 1986). By contrast, a 1987 assessment of computer use in Oregon public schools reported that word processing represented 17% of computer use in the schools surveyed. Moreover, 58% of teachers in those schools described the use of computers in language arts as "essential" or "important" (Lamon, 1988). Clearly teachers are finding a place for word processing in the language arts curriculum. Debate about computers in writing instruction no longer addresses *whether* to use them, but *how* to do so.

Much of this debate focuses on the merits of introducing formal keyboarding skills, including the prescribed "reaches" and postures of touch typing (Balajthy, 1988; Warwood, 1985; Suttles, 1983). Balajthy contends that the most often cited hindrance to the effective use of word processing is the lack of keyboarding skills (Balajthy, 1988). But exactly what keyboarding expertise do children need at various grade levels and how can they acquire that expertise? We are concerned that this focus on keyboarding skills may complicate rather than facilitate children's writing. We wonder how skillful children need to be in order to use word processing and what needs to be done to help them become skillful? Should young children learn correct fingering in order to avoid having to unlearn bad habits later (Balajthy, 1988; Schmidt, 1983) or is it sufficient to help children to locate the keys (Jackson & Berg, 1986). "Presently there is more opinion than solid research to guide decisions on the when and the why of teaching keyboarding" (Kroenke, 1987, p 244).

This paper addresses the issue of keyboarding instruction in elementary schools. Drawing from our research, we examine the arguments for such instruction in terms of children's abilities (the developmental issues) as well as the availability of time and equipment (the ecological considerations). We question whether the investment of time and effort is a worthwhile use of limited resources. A formidable investment of resources is required for children to master the techniques of touch typing: whole class instruction (one child per computer) including several thirty-minute sessions a week over at least a six-week period (Woodward, 1985; Milkes, 1985). We do not question that with an investment of time and equipment, children can be trained to be touch typists. We do, however, question whether touch typing is a necessary *prerequisite* skill for writing with computers in elementary schools

In addressing this issue we draw on data from two research studies developed under the aegis of the Literacy Research Center of the Graduate School of Education, University of Pennsylvania. One, "Microcomputers in Writing Development", examined the word processing and writing experiences of ten classes of children from kindergarten to fourth grade over three years. The other, "Cooperative Writing in Computer Environments" collected data over one year from fourth and sixth grade classrooms in a public school in Philadelphia where word processing was taught without whole class formal keyboarding instruction. Based on our findings we argue that insistence on touch typing skills is an exaggerated response to children's unfamiliarity with keyboards. We further argue that this issue is an artifact of an outdated vision of keyboarding, one that fails to take into account the ways children in schools actually use computers for writing. We question the effectiveness of such training as well as the merit of using school time and limited resources for it.

KEYBOARDING—AT WHAT AGE?

Keyboards are not straightforward tools: the letters are oddly arranged, while spacing and capitalization require specific keystrokes. In addition, word processing packages necessitate instruction in cursor movement as well as deletion and insertion of characters. If children are to use word processing, they need to know how to use the keyboard.

Touch typing skills have been taught in public schools since 1915 (West, 1983), often as an elective. Personal and business typing courses are offered in most secondary schools. The issue is not whether touch typing is a useful skill, but rather, as computers become available in schools, when to teach touch typing in the K-12 curriculum? We argue that touch typing skills are not prerequisite to computer use for writing. However the arguments against touch typing for primary age children are different from the arguments against touch typing for middle school age children.

WHAT OUR RESEARCH INDICATES

The Developmental Arguments

The argument against teaching touch typing to primary age children is based on important developmental issues. The prescribed reaches of touch typing were designed for adult hands, not the hands of young children. Young children (kindergarten and first graders) have enough to do to decide what to write, how words are spelled and what the next letter should be without worrying about which finger to use. Even if they know the associations for touch typing, their facility in composing will be more dependent on their ability to figure out spellings than on their ability to figure out the keys.

Repeated observations of kindergarten and first grade children in the "Microcomputers and Writing Development" project shows that touch typing was not necessary for these children. It was surprising to the research team, as they observed children working at computer keyboards, to find that these children needed very little instruction to get started with word processing. Keyboarding speed was not an issue, since they took as long to decide what letter to type as they did to find it on a keyboard. Children in these classes reported that typing was easier than handwriting. When they began to use the computers to write their own stories, using invented spellings or copying spellings from other sources, the facilitating qualities of word processing (e.g. automatic left-to-right and top-to-bottom progression, legible letters spaced evenly on the screen) were apparent to the children. The research team observed that in many cases children's writing with word processing was more ambitions than their writing with pencil and paper. One child composed this sentence "Yns opn a tm dr ys a hse" (Once kupon a time there was a house) with word processing in November in kindergarten. She made no such attempt with pencil and paper.

Older children -- second, third and fourth graders -- reacted somewhat differently to writing with computers. Familiarity with the keyboard was a factor for these children. In end-on-year interviews, second graders who had used the computers to write at least three separate pieces reported that they preferred writing with computers, while those who had less opportunity to use computers reported that they preferred to compose with pencil and paper, because they had difficulty finding the letters on the keyboard. In other words, children became familiar with the keyboard as they wrote, and the more they wrote, the more familiar they became.

Third and fourth graders all preferred to use word processing at some point in their writing. When asked what teachers should teach in order to help children use word processing, they said that children needed to know where the keys were. Their teachers had considered teaching touch typing. In fact they began each year by introducing the notion of correct fingering. However they abandoned this is favor of practice sessions with laminated printouts of the computer keyboard (Kahn, Avicolli & Lodise, 1987). Rather than stress the reaches of touch typing, they chose to familiarize the children with the locations of the keys, assuming that this familiarity would be sufficient. These practice sessions lasted five minutes each day for the first six weeks. (In contrast to elaborate requirements of whole-class instruction in touch typing, this activity could be done in spare time between other activities). In end-on-year interviews, almost every child commented on how helpful these practice sessions were. Observations and interviews with teachers during the three-year study made it clear that instruction of this kind in addition to writing with word processing allowed children to work easily at the computers.

The argument has been made that this sort pf practice fails to supply the kinesthetic experience of working at a keyboard, and therefore is not valuable. West (1983) addressing this same objection in relation to practice on different keyboards, argues that the advantages of practice on any keyboard outweigh the possible minor interferences for the feel of the stroking from one machine to another. He feels that the mental practice of finding the keys on any keyboard is also valuable because learning a keyboard is associative rather than response learning. It is precisely this activity that our students found helpful then they began to write with word processing.

Research conducted by Freyd as part of the "Cooperative Writing in Computer Environments" examined handwriting speed and typing speed in fourth and sixth grade classes where word processing was introduced for writing. Because of limited access to computers, the teachers in the study made the decision not to teach touch typing skills in formal lessons to the whole group. Instead they introduced the children to the idea of home keys and showed them how to use a computer typing program that was available for use during students' free time. In October the mean typing speed of 34 sixth graders who had had previous instruction in Logo was 6.62 words per minute.(as measured by the typing test on Keyboard Cadet, the typing program supplied by the district). Between October and May children received no formal typing instruction for speed or accuracy. Instead, they composed and revised in pairs at computers for approximately one hour each week. Occasional reminders were made to individual students to encourage them to use all fingers of both hands. Observations of students at computers showed that they moved in and out of touch typing techniques in relation to the writing task. When there was a rush to get words on paper, then the fastest method was used. When there was a lull between ideas, proper fingering was tried. By May, the mean typing speed had risen to 10.12 words per minute. This figure is interesting because it is not significantly different from the average handwriting speed of children of this age (8.8 - 12.5 words per minute -- Groff, 1961), nor is it significantly different from the average handwriting speed of these children (11.44 words per minute).

The argument has been made that for children to use word processing successfully, they need to be able to type at least as quickly as they write by hand (Fidanque, Smith & Sullivan, 1986). Wetzel (1985) suggests that the ability to type ten words per minute allows children to make adequate use of the computer and word processing programs. it would seem that keyboarding instruction lasting fifteen hours (Wetzel, 1985), sixteen to eighteen hours (Warwood, 1985) or thirty hours (Jackson & Berg, 1986) is not necessary. Freyd's findings indicate that this goal can be achieved as children write. It should be noted that typing speed gains that are achieved after intensive training are often lost if there is not consistent practice (Warwood, 1985). It appears that for elementary school children time may be better spent in production typing with steady slow gains over several years.

Freyd's research also shows a highly significant correlation between handwriting speed and typing speed ($p > .001$). That is, children who were able to handwrite quickly were able to type quickly. This finding suggests that it is inappropriate and probably unproductive to demand that all children acquire some minimum typing speed, any more than we require a minimum handwriting speed.

COMPUTERS IN SCHOOLS
The Ecological Arguments

Although computers are still a limited resource in elementary schools, we are sure that word processing is a valuable tool that can facilitate children's composing. Our research has shown that children learn fairly quickly to locate the keys, and that they can do so with limited instruction or with repeated opportunities to write. The ecological argument against elaborate keyboarding instruction in elementary schools takes into consideration the environment in which teachers and children work and the resources of time, equipment and expertise available to them.

In elementary schools, where the student/computer ration is still more than thirty to one (Office of Technology Assessment, 1988) it is difficult to provide adequate access to computers. The curriculum is full and resources are scarce. Given this reality, teachers must decide how best to use computers. All of the teachers in the "Microcomputers and Writing Development" study, given autonomy to make curricular decisions concerning the use of word processing, chose to focus on writing rather than typing skills. however, even with this commitment to computers in writing, the lack of equipment (two computers per room, twelve students per computer) and pressures of other curricular demands meant that some children only used the computers three times during the school year. If these teachers had felt compelled to teach touch typing, the children might never have used the computers for writing at all. The teachers in the "Cooperative Writing" study also felt that limited time was better spent teaching block movement commands to aid revision than teaching typing skills.

It will be noted that the teachers in the "Cooperative Writing" study worked with their classes in a lab setting (two children per computer) while the teachers in the "Microcomputers and Writing Development" study had two computers; in each classroom. The location of computers in elementary schools is a central issue. When asked for their recommendations concerning the distribution of computers in the school, the "Microcomputers" teachers unanimously opted for computers in their rooms, rather than in a lab. They felt that they had greater flexibility in scheduling children at the computers, and the opportunity to use the computers for other activities, such as making banners and taking spelling tests. The teachers in the "Cooperative Writing" study initially had no choice concerning the distribution of computers, but have since successfully applied for classroom computers. They also valued the increased flexibility in scheduling that is possible when there is a computer in the classroom.

Even when computers are not a limited resource, time is. It is difficult to find thirty or forty-five minutes three times a week for six weeks in the elementary school schedule, a time commitment that seems to be the minimum required for mastery of keyboarding skills. This intensive practice is only the beginning requirement of touch typing advocates. Balajthy (1988) admits that the gains in speed and accuracy attained with these sessions will be lost if children do not have continuous access to computers. In other words, setting out this block of time for keyboarding skills is not enough; children need to consolidate their typing skills by using them for writing tasks.

Our argument is that children can learn to type within meaningful writing activities rather than with concentrated practice on unrelated drills. Other researchers have found this to be true. Barton (1926, as reported in West, 1983) reports that college students who learned the keyboard by composing simple business letters on themes set by the instructor did better at the end of the semester than those using nonsense drills for the first nine weeks to learn the keyboard. Commenting on this sixty-year old research finding, West observed that one of the truisms about formal schooling is that twenty-five to fifty years generally elapse between the first appearance of superior practice and its general adoption in classrooms. It may be just about time to let children learn to type within meaningful writing rather than nonsense drills.

The ecological argument we make is that time, equipment and expertise in elementary schools are better used to help children compose and revise on keyboards beginning in kindergarten. To wait until some preliminary level of keyboarding skill has been attained unnecessarily postpones the very real benefits of writing with word processing. No one suggests that children postpone writing by hand until they master cursive writing skills. Likewise, keyboarding skills should not be a prerequisite to writing with computers.

KEYBOARDING SKILLS

To what end?

Why are keyboarding skills an issue now? We believe that this issue is an artifact of an outdated vision that has more to do with secretaries and typewriters than with computers and children. Salomon & Gardner (1986) observe that new media begin by adopting the contents and formats of the media they are likely to replace or modify. Weizenbaum (1976, p. 32) comments that it is rare for "a tool and an altogether original job it is to do, to be invented together." Such is the case for word processing in elementary schools. We have transferred the methodology that came with typewriters to computers when in fact the new technology is different from the old in important ways, and those ways make the old methodology irrelevant. In addition, the ways children in elementary schools use word processing are different from the practices of secretaries in offices.

The methodology of typing instruction which uses nonsense syllables and emphasizes speed may even be irrelevant in traditional typing classes. West (1983), in a comprehensive text, *The Acquisition of Typing Skills*, states that concern with keystroking skills is justified if a) straight copy typing is a reasonably frequent activity and b) stroking skills make a significant contribution to typing proficiency. He goes on to make a distinction between copy typing and production typing, which includes composing at keyboards. If the arguments can be made for a typing class, they certainly apply to writing instruction in elementary schools. "Production typing adds as much (or more) to stroking skills as does direct practice at them" (West, p. 186). In other words, even in typing class, using keyboards for real writing tasks is as valuable as keystroking exercises. And what are the skills that keystroking exercises are designed to build? The ability to type with accuracy, speed and uniform pressure on the keys without looking at the keys. But are these skills relevant to the ways children use computers for writing?

Let us first consider the issue of accuracy. Both teaching methods and electronic media suggest that accuracy is not an issue when young children compose, particularly first drafts. The most recent research on children's composing points to the value of postponing concerns about correct spelling, grammar and punctuation until meaning has been clarified (Graves, 1982). Furthermore, accuracy is an issue when it is difficult to make corrections, as it is on paper with pencil. Making corrections on a computer screen is literally child's play. Research has shown that children enthusiastically edit their work with word processing, attributing their enthusiasm to the ease of correcting on an electronic screen rather than on paper (Kahn 1987).

Is speed a concern when composing with computers? Composing is not transcribing; it requires thought and reflection. And composing (which West includes in production typing) is what children in elementary schools do. Word processing is valuable precisely because it simplifies the production of text, freeing children to concentrate on creating meaning. Our research shows that children only need to type at least as quickly as they write -- ten words a minute -- in order to compose as comfortably at computers as they do on paper. We know, because we have asked the children, that when they can find the letters on the keyboard they all report that it is easier to push a button than write a letter.

Uniform pressure on the keys is truly an artifact that has nothing at all to do with electronic screens and printers. The final product will be perfectly printed no matter how unevenly children press the keys. The only thing that children need to learn about pressure on keyboards is not to press too hard, because letters will be repeated on the screen. Beginners only need to see this happen once to learn the appropriate degree of pressure.

> "Typewriting appears to be the only skill on the face of the earth taught to sighted person as if they were blind" (West, p. 46).

Do children need to be trained not to look at the keys? Secretaries doing copy typing can not look at the copy and at the keys simultaneously. This is the reason they are taught touch typing. But composing at keyboards is a different issue entirely. We have seen rooms full of professional programmers composing at keyboards. These programmers continuously watch the keys and have developed reasonably quick hunt-and-peck strategies. For beginners at keyboards at any age, West (1983) suggests that looking at the keys helps students to learn their locations.

In a related argument, teachers express concern that using keyboards without touch typing skills will reinforce "bad habits" that will be difficult to unlearn. West (in a personal communication) states that twelve hours of formal touch typing instruction is sufficient to undo any "bad habits" and that, as our own experience suggests, familiarity with the locations of the keys facilitates learning touch typing. Children manage to learn cursive writing after we teach them quite different printed letter forms. Our point is that many students who have been hunting and pecking for years successfully learn touch typing in high school when they are motivated to do so and where it is easy to find time, equipment and a competent teacher.

CONCLUSION

We return to the questions with which we began this paper. What do children need to know in order to use word processing for writing? Children need to know where the keys are, including the space bar, return key and punctuation marks, and how to use cursor movement arrows or commands to insert and delete characters With this information, children find word processing to be a "convivial tool" (Eisner, 1985).

Our paper refutes the claim that touch typing instruction is necessary for elementary school children. Instruction in touch typing is an exaggerated response to young children's unfamiliarity and lack of expertise with the keyboard. It is true that children are often frustrating slow at the keyboards when they begin to use computers. Our research shows that by the time children have used the computers for writing on three separate occasions, they have become sufficiently familiar with the locations of the keys to use word processing easily. Any further learning about the keyboard can take place as they do meaningful writing.

An underlying assumption of teaching touch typing may be that keyboarding difficulties interfere with production of text and that the ability to type quickly and accurately might remove that interference. A related assumption is that to get one's thoughts on paper quickly would facilitate writing and that lack of speed in transcription causes a writer to lose his or her train of thought. The value of introducing computers and word processing, however, has to do with their ability to reduce the interference factors that might affect production in writing -- i.e. to make writing easier and less tedious. The operative word is easy -- Frank Smith (1987) marvels that young children ever write anything at all with pencils and paper -- and children describe in lurid detail the trials and tribulations related to paper and pencil usage. What needs to be done to make writing physically easier for children?

Research indicates that for young children, physical production of text claims attention and therefore interferes with the ability to compose (Scardamalia & Bereiter, 1982; Graves, 1982). Advocates of word processing in schools argue that computers for composing can circumvent such issues as the formation of letters, left-to-right and top-to-bottom progression, and motor ability (Smith, 1987). This may be so for primary age children, but considering the difficulties these children often have with handwriting, it takes very little exposure to a computer keyboard for a child to type more quickly (and more legibly) than he or she can write.

For middle school age children, however, composing at keyboards is not simply a matter of producing text quickly. Children of this age suffer less than younger children from the interference of factors related to production, and research tells us that removing the interference of production factors does not improve the quality of their work. Gerlach (1987) presents evidence that concentrated keyboarding instruction (fifteen lessons, each twenty-five minutes long) does not lead to either longer compositions or more extensive revisions. Word processing makes writing easier t=for these children, but that is all. We feel that classroom instruction, response groups and regular publication of written work will have a greater impact on the length and quality of children's writing than the acquisition of keyboarding skills.

Our research indicates that with brief introductions to the locations of the keys and encouragement to use both hands, children learn to manage computer keyboards as they do their writing. This ability to produce text easily allows children to concentrate on composition. The introduction of touch typing, where students must type without looking at the merely substitutes new production difficulties for old. Furthermore, we risk losing sight of the objective. In elementary schools word processing should facilitate composing rather than introduce a new set of problems.

Bibliography

Balajthy, E. (1988). Keyboarding, language arts, and the elementary school child. *The Computing Teacher*, (Feb), 40-43.

Becker, H. J. (1986). Instructional Uses of School Computers: Reports from the 1985 National Survey. Center for Social Organization of Schools, Johns Hopkins University.

Bereiter, C. & Scardamalia, M. (1987). *Psychology of Written Composition*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Eisner, E. (1985). *The Educational Imagination*. NY: Macmillan, Inc.

Fidanque, A., Smith, M. & Sullivan, G. (1986). Keyboarding: The issues today. *Proceedings '5th*

Annual Extending the Human Mind Conference . University of Oregon, 1986.

Gerlach, G. (1987). The effect of typing skill on using a word processor for composition. Paper presented at Annual Meeting of the American Educational Research Association (Washington, D.C.).

Graves. D(1984). *A researcher learns to write.* Exeter, NH: Heinemann Educational Books, Inc.

Jackson, T. & Berg, D. (1986). Elementary keyboarding - is it important? *The Computing Teacher*, 13, (Mar), 8-11.

Kahn, J. (1987) Emergent writers using word processing: Making it easier and what that means. Paper presented to the Eastern Educational Research Association, Miami Beach, FL.

Kahn, J., Avicolli, M. & Lodise, K. (1988). Learning the keyboard. *Interface* 4(4) (Jan).

Kroenke, K. (1987). Keyboarding: Prelude to composing at the computer. *English Education* (Dec), 244-249.

Lamon, W. (1988). Using microcomputers in the elementary schools: The 1987 Oregon assessment. *SIG Bulletin for Teacher Educators*, 4(2). Eugene, OR: International Council for Computers in Education.

Milkes, J. (1985). *Teacher's Manual for Keyboarding Kapers*. Bronx, NY: MPC Educational Publishers.

Salomon, G. & Gardner, H. (1986, Jan.). The computer as educator: Lessons from television research. *Educational Researcher*, pp. 13-19.

Scardamalia, M. & Bereiter, C. (1982) The role of production factors in writing ability. In M. Nystrand (Ed.). *What writers know : The language , process, and structure of written discourse (pp. 173-210).* Orlando, FL: Academic Press, Inc.

Power On!: New tools for teaching and learning. Congress of the United States Office of Technology Assessment SET-380. September 1988.

Schmidt, B. (1983). *Keyboarding: the state of the art: Project report.* Blacksburg, VA: Virginia Polytechnic Institute and State University. [ED 236 352].

Smith, F. (1987). The promise and threat of computers. *Insult to Intelligence*, New York: Manor House.

Suttles, A. (1983). Computers and writing: Contemporary research and innovative programs. *Computers in Reading and Language Arts ,1*, (Summer), 33-37.

Warwood, B., Hartman, V., Hauwiller, J. & Taylor, S. (1985). A research study to determine the effects of early keyboard use upon student development in occupational keyboarding. Bozeman, MT: Montana State University. [ED 265 367].

Weizenbaum, J. (1976). *Computer power and human reason: From judgment to calculation.* San Francisco, CA: W. H. Freeman and Co.

West, L. (1983). *Acquisition of typewriting skills.* Indianapolis, IN: Bobbs-Merrill, Inc.

Wetzel, K. (1985). Keyboarding skills: Elementary, my dear teacher? *The Computing Teacher*, 12, (June), 15-19.

THE EFFECTS OF COMPUTER ASSISTED INTERACTIVE FICTION ON THE DEVELOPMENT OF READING COMPREHENSION IN GRADE FIVE STUDENTS

Robert F. Whitelock
Bruce County Board of Education
Bruce Peninsula District School
Lion's Head, Ontario, Canada
N0H 1W0

Abstract

Forty-five grade five students from a small city in northern Canada were divided into experimental and control groups. Students in the experimental group engaged in fifteen minutes of computer assisted interactive fiction each day over an eight-week period. Students in the control group were involved in uninterrupted sustained silent reading for an equal period of time. The students' gains in reading comprehension were tested using forms A and B of the Stanford Diagnostic Reading Test (Brown Level).

It was found that students engaging in computer assisted interactive fiction gained significantly greater than the control group in the structural analysis sub-skill, but not in overall reading comprehension. It was also found that students using text adventures of higher reading difficulty had significantly greater gains in inferential comprehension than those working with lower difficulty texts. Overall there was no significant difference found between the comprehension gains of students using computer assisted interactive fiction and those engaging in traditional sustained silent reading. Computer assisted interactive fiction may be as useful as sustained silent reading in developing reading comprehension.

This study yielded data which was problematic at best. This may have been a result of the short experimental period, or may reflect the low re-test reliability of the Stanford Diagnostic Reading Test. It does point out the need for the classroom use of these readily available commercial products to be critically examined in further research studies.

Interactive fiction refers to stories which are written in such a way that the reader is able to direct the nature of the story being read. Those which have appeared in print form use a convention wherein the reader is asked to make decisions on behalf of a character in the story, and select a path for that decision by turning to a pre-selected page in the story, where the results of the decision are discovered. The theory behind such writing is that children will be more interested in reading if they are personally involved in the story. The drawback, in printed form, is that the stories tend to be very brief, in order to prevent the book with its multiple branches from becoming too large.

Computers provide a means whereby these stories can be presented in great detail, as the information is stored on disk and therefore lacks the bulk of paper-based versions. In the computer context, interactive fiction has much more breadth and depth than is possible in printed form. In addition, the computer can be made to respond to the actions of the user immediately, without page flipping. A wide variety of interactive fiction has been available for microcomputers since the early 1980s, written at a variety of levels of reading difficulty.

The purpose of this study was to determine the effects of regular student involvement in computer based interactive fiction on the development of reading comprehension in grade five students. It was hypothesized that student use of computer based interactive fiction would not result in significantly different gains in either overall reading comprehension or reading comprehension sub-skills compared with students involved with uninterrupted sustained silent reading (USSR) for equal periods of time, and that there would be no difference in overall or sub-skill reading comprehension gains due to student ability.

Background

The mechanism whereby children learn to understand the meaning of printed words is not well understood (Nicholson, 1986). Factors which seem to be involved include prior knowledge (Rumelhart, 1980), student interest (Good and Brophy, 1980; Asher, 1980), and time given to silent reading (Leinhart, Zigmond and Cooley, 1981).

A popular method of providing students with practice silent reading time has been the adoption of USSR programs. Students are required to spend a set period of time each day reading material of their choice. This method is designed to use topic interest as a vehicle for developing interest in reading, and it tends to be successful, although the impact on reading comprehension is not clear (Manning - Dowd, 1985; Schaudt, 1983).

The vast majority of computer related reading activities involve simple drill-and-practice or tutorial exercises (Geoffrion and Geoffrion, 1984/85). These can develop reading sub-skills, but are not particularly suited to holistic approaches to the development of reading skills. In effect, most computer assisted instruction related to reading are merely automated paper and pencil exercises.

Computer assisted interactive fiction, commonly referred to as text adventure games, involves the reading of large amounts of text on the part of the student, rather than simple drills. Students playing text adventures tend to be very motivated to continue the story, even when difficulties arise (Lancy and Hayes, 1986). As a result, a number of theorists have suggested the use of computer assisted interactive fiction as an alternative activity for students (Grabe and Grabe, 1985; Little, 1985; Brady, 1986; Jarchow and Montgomery, 1985; Layton, 1987).

Method

The sample consisted of 45 students (26 male, 19 female) from two fifth grade classes in one school. They ranged in age from 10 years 1 month to 12 years 6 months. Subjects were assigned randomly to the experimental and control groups.

The experiment followed a true experimental pre-test post-test design. All subjects were pre-tested for ability with the Canadian Cognitive Achievement Test, achievement with the Canadian Test of Basic Skills, and reading comprehension using the Stanford Diagnostic Reading Test, Form A. Following the treatment period, all students were post-tested for reading comprehension using the Stanford Diagnostic Reading Test, Form B.

The experimental treatment consisted of an eight week period in which the experimental group played a text adventure game for fifteen minutes per day, every day, on an Apple //e microcomputer. The experimental subjects had their choice of four Infocom text adventures which were rated for beginners, and were able to switch choices during the first week. The comparison treatment, which the control group followed, involved uninterrupted sustained silent reading at the same time as the experimental group was engaged in interactive fiction.

Results

The experimental and control groups were statistically compared with respect to ability, achievement, sex, age, reading text difficulty and attendance. Of these, the only significant difference between the groups was found in the difficulty of the text being read ($t = 4.244$, $p <= 0.05$). As a result, the subjects involved in the two groups were considered to be equivalent.

The experiment was designed to consider four null hypotheses. These included mean gains in overall reading comprehension and mean gains in reading comprehension sub-skills, comparing experimental versus control groups and ability groups within the experimental group. Of the four only the second, which predicted no difference between the groups on the basis of any of the reading sub-skills measured by the Stanford Diagnostic Reading Test was rejected. The hypotheses and the results are listed in Table 1.

In order to comprehend the meaning of these results, it was necessary to take a more detailed look at the data. Measured information was available reflecting sex, age, and ability; achievement in vocabulary, reading, spelling, and overall language; attendance and reading text difficulty. These variables were examined against the gain scores provided by the Stanford Diagnostic Reading Test. Gain scores were available in overall reading comprehension, as well as phonetic analysis, structural analysis, auditory vocabulary, literal comprehension and inferential comprehension sub-skills.

The groups were first divided into high and low portions in each variable, and the high portion of the experimental group was compared to the high portion of the control group. This was repeated for the low portions, and for each variable. The resultant t-tests of these comparisons are contained in Table 2.

Examination of the t-tests shows that in the majority of cases, a significant difference was not found between the experimental and control groups. Of significance was the fact that most of the places where a significant difference was found was with respect to the structural analysis sub-skill, in which the experimental and control groups were significantly different in age, ability, vocabulary, spelling, total language and text difficulty. A positive t-test result in the table indicates gains in the experimental group exceed those for the control group. This supports the finding in favour of rejecting the second null hypothesis.

The only other finding from the examination of t-tests was related to the text difficulty variable. In this study text difficulty refers to the reading level of the texts. On the basis of high and low text difficulty, the control students with low difficulty texts exhibited significantly greater gains than those students reading lower difficulty text adventures in inferential and overall reading comprehension, while the experimental group with low difficulty text adventures gained more than their counterparts in structural analysis. This odd result may be due to the fact that the control group read materials of lower difficulty than the experimental group.

Table 3 contains a representation of the comparison of the high and low portions of the experimental and control groups within themselves. It can be seen from this table that the high portion of the experimental group achieved significantly higher gains than the low portion in inferential comprehension across most of the variables. This pattern was reversed somewhat in auditory vocabulary, where students with lower achievement exhibited higher gains than the high group amongst the experimental group. In each case, the pattern was only partially supported within the control group. As a result, it would appear that the treatment had an impact on the experimental group with respect to auditory vocabulary and inferential comprehension.

In addition to the measured data, it was found through observation of the experimental group that involvement in interactive fiction had a strong positive effect on the willingness of individual students to stick to the games and try to complete them, regardless of whatever problems they encountered. Several students had repeated difficulties in getting through their story, and no student completed the adventure game in the allotted time. Despite this, however, no student ever complained about the exercise, and most were upset at the end of the experimental period that they could not continue past the eighth week. One student actually wrote his/her own paper-based text adventure as part of a language arts writing assignment. Another, a special education student who had the lowest ability measurement of the subject population, worked repeatedly at his/her text adventure, and was able to complete the testing at the end despite not having answered any reading comprehension questions on the pre-test of reading comprehension.

In summary, the study found that students engaged in computer assisted interactive fiction had higher gains in the structural analysis sub-skill than those engaged in uninterrupted sustained silent reading. In addition, students of the experimental group with higher difficulty text had higher gains in inferential comprehension than those with lower difficulty texts. Overall, there appeared to be no significant difference between computer based interactive fiction and uninterrupted sustained silent reading in the development of reading comprehension.

Discussion

At first glance it may appear that there is no merit in the assertions of Jarchow and Montgomery (1986) and Layton (1987) with regard to the potential of computer assisted interactive fiction in classroom instruction. Careful examination of the data, however, indicates significant problems with the gain scores. The average standard deviation of gain score results obtained from the Stanford Diagnostic Reading Test was over nine times greater than the comparable mean gain score. This implied that there were problems in the comparative data available.

These problems were traced to several potential sources, including the test/retest stability of the subjects, variability in the conditions surrounding the testing, test scoring, and the comparability of the two forms of the Stanford Diagnostic Reading Test. Examination of all of these possibilities concluded that the most likely source was the alternate form reliability of the Stanford Test. This problem may have been reduced or eliminated if the subjects had been involved in a longer experimental period.

While the results of this study must be questioned due to the problems in the data, the results need not be discarded. The consistency of the range of scores across groups and tests, combined with the strength of the significance in the two cases that were found overall to be different from their comparison groups, suggests that the results may be replicable. In addition, the results do not suggest that computer assisted interactive fiction was less valid than USSR in the development of reading comprehension. Teachers could therefore feel reasonably secure in substituting computer text adventures for more conventional recreational reading.

This study is in need of replication. To improve the reliability of the data gathered, replication should include a much longer experimental period. In addition, a second study should assess the impact of the games on student interest in reading as well as reading comprehension, as informal observations of this study suggested text adventure playing may have very positive effects. It would also be important to ensure that the instruments used to assess gains have extremely high alternate form reliability.

This study does not establish the superiority of computer assisted interactive fiction in developing student reading comprehension, but it does serve to demonstrate the need for further research into the use of these games as a classroom reading tool.

Acknowledgements

This study was undertaken through the Department of Graduate Studies and Research, College of Education, Brock University at St. Catharines, Canada. The assistance of Dr. A. Wheeler, and of Dr. G. Samson of Cleveland State University are sincerely appreciated. The study was conducted with the assistance and participation of employees and students of Fort McMurray School District 2833, Fort McMurray, Alberta, and with the support of the Bruce County Board of Education, Bruce County, Ontario. The willingness and assistance of all involved is gratefully acknowledged.

References

Asher, S.R. (1980). Topic interest and children's reading comprehension. In Spiro, R.J., Bruce, B.C. & Brewer, W.F. (eds.). *Theoretical issues in reading comprehension.* Hillsdale, N.J.: Lawrence Erlbaum Associates. 525 - 534.

Brady, P. (1986). Computer simulations and reading instruction. *The Computing Teacher, 13* (2), 34 - 36.

Geoffrion, L.D., & Geoffrion, O.P. (1984/85). Comprehension. *Computers in Reading and Language Arts, 2,* 18 - 24.

Good, T.L., & Brophy, J.E. (1977). *Educational psychology: A realistic approach.* Toronto: Holt, Rinehart and Winston.

Grabe, M., & Grabe, C. (1985). The microcomputer and the language experience approach. *The Reading Teacher, 38,* 508 - 511.

Jarchow, E., & Montgomery, J. (1985). Dare to use adventure games in the language arts classroom. *English Journal, 74,* 104 - 106.

Lancy, D.F., & Hayes, B.L. (1986). *Building an anthology of interactive fiction.* Paper presented at the annual meeting of the American Education Research Association, April 16 - 20, 1986.

Layton, K. (1987). Interactive text adventures. *Media and Methods*, (March/April), 14 - 16.

Leinhardt, G., Zigmond, N., & Cooley, W.W. (1981). Reading instruction and its effects. *American Educational Research Journal, 18*, 343 - 361.

Little, D. (1985). Adventure games in the curriculum: Problems and potential. *ERIC Document Reproduction Service* ED 265 491

Manning-Dowd, A. (1985). The effectiveness of SSR: A review of the research. *ERIC Document Reproduction Service* ED 276 970

Nicholson, T. (1986). Reading is not a guessing game: The great debate revisited. *Reading-Canada-Lecture, 4*, 239 - 246.

Rumelhart, D.E. (1980). Schemata: The building blocks of cognition. In Spiro, R.J., Bruce, B.C. & Brewer, W.F. (eds.). *Theoretical issues in reading comprehension*. Hillsdale, N.J.: Lawrence Erlbaum Associates. 33 - 58.

Schaudt, B.A. (1983). Another look at sustained silent reading. *The Reading Teacher, 36*, 934 - 936.

Table 1
Summary of Hypotheses and Decisions

Hypothesis		Decision	Basis
1	Experimental group mean gains in reading comprehension no different than gains of control group.	Accept	t = -0.684 not significant.
2	Experimental group mean gains in each reading comprehension sub-skill no different than mean gains of control group.	Reject	Structural Analysis t = 1.431 (p<=.1)
3	Subjects in experimental group will not differ by ability in mean gains in reading comprehension.	Accept	F = 1.128 not significant.
4	Subjects in experimental group will not differ by ability in mean gains in any sub-skill of reading comprehension.	Accept	F ratios not significant.

Table 2
Comparison of T-test Results Comparing Above Average Experimental and Control Group Subjects and Below Average Experimental and Control Group Subjects

	Group	Age	Ability	Vocab	Read.	Spell.	Lang.	Text Diff
Phonetic	H	.744	-.247	-.567	-.276	-.457	.177	.247
	L	-1.364*	-1.321	-.209	-.879	-.357	-1.889**	-.635
Structural	H	-.247	2.016**	-1.548*	1.201	.19	.793	.837
	L	1.875**	.232	-.394	.766	1.978**	1.589*	1.618*
Auditory	H	.137	.185	1.125	.619	.866	.598	.028
	L	.352	-.485	-.569	.052	-.076	-.048	1.206
Literal	H	-.144	-.722	-.023	.086	-.385	-.217	.418
	L	-.645	-.052	-.977	-1.049	-.507	-.829	-.461
Inferential	H	-.031	.526	.257	.866	.453	-.137	.644
	L	-.172	-.133	-.568	-1.33	-1.007	-.294	-2.19**
Total Comp	H	-.172	-.237	-.007	.425	.014	-.336	.58
	L	-.569	-.052	-.855	-1.298	-.919	-.636	-1.694*

Key:
H t-test comparing higher than average E and C groups
L t-test comparing lower than average E and C groups
* $p <= 0.1$
** $p <= 0.05$

Table 3
Comparison of Reading Comprehension and Reading Sub-Skills by Variables Yielding Statistically Higher Gain Scores

	Group	Sex	Age	Ability	Vocab	Read.	Spell.	Lang.	Text Diff	Attend
Phonetic	E									
	C									
Structural	E	L	H	H					L	
	C									
Auditory	E			L		L		L		
	C	M	H	H	L	L	L	L		
Literal	E			L						
	C	M							L	
Inferential	E		L	H	H	H	H	H	H	
	C	F	L				H	H		
Total Comp	E		L			H	H			
	C		L						L	

Key:
E experimental group
C control group
M males
F females
H higher than average
L lower than average

TOOLS FOR MESSY MATHEMATICS: USED NUMBERS AND ELASTIC

Andee Rubin
BBN Systems and Technologies Corporation
10 Moulton Street
Cambridge, MA 02138-1191

Susan Jo Russell
TERC

ABSTRACT

Mathematics is often thought of as a pure science, a precise framework that avoids the inaccuracies and vagaries of the physical world. Doing mathematics (and thus learning mathematics) is conceived of as working with ideal objects, producing an elegant proof from beginning to end, finding and following the right rules, arriving at the correct answer. Recently, this stultified view of doing mathematics has been challenged by researchers who, instead, characterize mathematics as an exploration, complete with hunches, hypotheses, evidence, and garden paths (e.g. Lampert, Schoenfeld, and Greeno)—a "messy" process.

We want to talk about the kind of mathematics that is inherently messy because it is derived from and intimately connected to real world situations. In particular, we will focus on mathematical activities that deal with data, because in collecting, organizing, displaying, and analyzing data, students come face-to-face with the messiness of the real world. At first, students—and even teachers—may be daunted by the difficulty of translating real world situations into the formal language of mathematics, but experience with data analysis can give them the tools to reach reliable conclusions in spite of the complexities of the world.

This session will describe two projects which use computer tools to support students' growing competence with data. The first project, Used Numbers: Collecting and Analyzing Real Data, is designed for students in kindergarten through sixth grade. It provides carefully-sequenced activities that introduce students to basic processes of data collection and analysis—counting, classification, estimation, measurement, graphing, and statistics—and then provides real data analysis situations which challenge students to use these ideas in solving problems which are of interest to them. For example, a third grade class investigated body measurements, leading to an analysis of the sizes of chairs needed in their classroom. A sixth grade class studied play injuries and was able to make recommendations about improved safety after school.

Computers and calculators are integrated into these activities for calculation, for organizing data, and for representing data. In particular, database and graphing software provide important tools throughout the units. A piece of LOGO-based software developed by the project, the Dynamic Bar Grapher, allows students to enter numerical or categorical data directly into a graph without first needing to organize it in a table or list. By allowing students to view their data in a variety of ways, such tools encourage them to *use* their data to construct interpretations, make predictions, and develop theories based on the information they have collected.

The second project, Visual Data Bases (VDB), is being developed as part of the ELASTIC project (Environments for Learning Abstract Statistical Thinking) at BBN Labs. Field tests of the ELASTIC software and curriculum showed that, although real world data collection activities are both motivating for students and provide experience with crucial skills of coping with messy data, logistical problems sometimes make such activities unpopular with teachers. The goal of Visual Data Bases is to create a computer-based data-collection environment that combines aspects of the real world with features that are responsive to the requirements of classrooms. A VDB provides students with a population of visual data, along with computer tools for measuring a variety of visual features. Just as in the real world, students must decide how to choose a sample, which features to measure, and how much precision to demand from their measurements. The computer, however, can facilitate the process by automatically recording measurements and organizing them into a spreadsheet as well as keeping track of the sample so that the data-collection process can be repeated if there are doubts about its accuracy. The first such VDB we have developed is a Hypercard stack of maps of the 50 United States, with which students can use computer-based tools to measure lengths (e.g. perimeter or coastline) and areas on each state map. A variety of other measurement tools (e.g. density and angle) are under development.

Visual Data Bases can also significantly extend student's data-collection opportunities. Hypercard stacks can include collections of images (e.g. daily weather maps from the last two years) to which students would otherwise not have access, but interactive videodiscs can open up even more exotic data-collection worlds for students. Weather maps can be complemented by video stills of cloud formations or sequences showing weather movement. Topics such as animal behavior and traffic patterns that require data collection over time can be investigated using "timer" tools to take measurements on a videodisc. To study these possibilities, we are currently developing a sample videodisc with an expanded set of measurement tools that will include a variety of visual data bases in areas such as weather, anthropology, ethology, and astronomy.

HYPERCARD AND HYPERTEXT

Barbara Dubitsky, Rosemarie Buzzeo
Michael Land, Sandra Turner
Randall Boone, Kyle Higgins

ABSTRACT: HyperCard: A Tool for the Mathematics Teacher

Barbara Dubitsky, Chair Computer Programs/Director Mathematics Leadership Program; Rosemarie Buzzeo, Mathematics Science Specialist, School for Children, Bank Street College of Education, 610 W. 112th Street, New York, NY 10025.

The presentation describes a pilot study in which mathematics software was produced by educators, who were not expert programmers, using HyperCard. The software was used as an integral part of the mathematics curriculum in the middle and junior high school in the Bank Street School for Children.

Before HyperCard was available, we held a strong conviction that teachers did not have the time to create computer programs that would be useful educational tools for their classrooms. Most of the teacher-produced, non-commercial software we had seen was not educationally viable. But when we experienced the ease with which ideas could be translated into software using HyperCard we wished to reexamine our notion.

A mathematics/science teacher in Bank Street College's children's school, and a mathematics/computer person in the graduate programs division of Bank Street wrote the software and tested it with children. Among the subjects the software dealt with were the concept of average, estimation in division, and two different approaches to algebra problem solving. It was found that the software provided help in learning mathematics that was unique to this medium. The children enjoyed using the material and asked specifically to use each piece again after it had been presented.

It is clear that HyperCard can be a useful tool for producing mathematics teaching material. We found, however, that, even with HyperCard, a great deal of time and energy are required to produce viable software.

ABSTRACT: HyperCard Ideas for the K-12 Curriculum

Michael Land, Sandra Turner, Department of Computer Education, National College of Education, 2840 Sheridan Rd., Evanston, IL 60201.

When HyperCard was first introduced, educators at all levels immediately recognized its value as an instructional tool. Teachers can use HyperCard at different levels of sophistication: to use stackware created by others, to customize existing stackware, or to design and develop professional-looking stackware on their own without learning a sophisticated programming language. HyperCard's ability to incorporate graphics and sound and to control CD-ROM disks is impressive. Most importantly, however, HyperCard provides a new environment for learning, an environment in which learners need not follow a predetermined linear sequence but can branch as they wish to follow up on a specific idea.

The purpose of this session is to demonstrate specific ways that HyperCard can be used to teach topics in the K-12 curriculum. The presenters have developed a collection of HyperCard stacks that illustrate the power of HyperCard as an instructional tool at the elementary and secondary levels. In addition, the strengths and limitations of HyperCard as an educational tool will be discussed.

ABSTRACT: Hypertext CAI: A Supplement to an Elementary School Basal Reading Program

Federal Grant Award Number H024J80015
Thomas C. Lovitt, Principal Investigator

Randall Boone, Kyle Higgins, University of Washington, Experimental Education Unit, C.D.M.R.C. WJ-10, Seattle, WA 98195.

This presentation will discuss the first year of a three-year federally funded project involving the development and testing of hypertext computer assisted instructional (CAI) reading materials. These materials are designed to facilitate the successful participation of both handicapped learners (learning disabled, emotionally disturbed, and educable mentally retarded) and nonhandicapped learners in a regular elementary classroom basal reading program (K-3). The hypertext reading lessons provide students with a reading environment offering additional information about words and concepts from their basal reading textbook. This differs from traditional computer assisted reading software which often presents lessons on isolated skills not specifically related to basal textbook content or pedagogy.

The hypertext CAI lessons are being developed in three levels of increasing sophistication. Over the three years of the project the lessons will cover three levels of instructional features: (a) level-one, text with enhanced surface structures; (b) level-two, text with enhanced syntactic and semantic structures; and (c) level-three, text with enhanced deep-meaning (comprehension and self-monitoring) structures.

Software developed for the first year of the project includes surface level enhancement to the basal reader text. The emphasis at this level is to provide as much additional information as possible for the student to access through the hypertext format. Related graphics, animated graphic sequences, simple definitions, synonyms, and digitized speech, linked to words and pictures from the original basal text are included to provide the students with additional experiential information.

ETHICS IN THE INTRODUCTORY COMPUTING COURSE

Ronald E. Anderson, Chair
University of Minnesota
Minneapolis, MN

ABSTRACT

Authors of major texts for introductory college computing courses will discuss their philospohy of ethics education. Panelists will respond to hypothetical ethical dilemms of typical students and instructors on such issues as sharing software, illegal entry, responsibility for software errors, and private use of institutional computing resources. The costs of including or excluding ethics in the classroom in various ways will be addressed. The panel will focus upon present and future ethical issues related to instruction in computing.

PANELISTS

Steven L. Mandell
Marsall and Melhorn Law Firm, and Bowling Green State University
Toledo, OH

Daniel D. McCracken
City College, City University of New York
New York, NY

Claire McInerney
College of St. Catherine
St. Paul, MN

Judith A. Perrolle
Northeastern University
Boston, MA

SPONSOR: ACM/SIGCAS

TECHNOLOGIES FOR SPECIAL NEEDS CHILDREN

Trisha Ainsa
Pamela Campbell
Antonia Stone, Tom Harris, Arthur Wood

ABSTRACT: Computers: A Literacy Connection for Young Children At-Risk

Trisha Ainsa, University of Texas at El Paso,1020 Broadmoor, El Paso, Texas 79912.

For the past five years, LEP (Limited English Proficient) children enrolled in pre-kindergartens, Head Starts, and kindergartens have been learning English through the use of computer concepts and enhanced developmentally appropriate curriculum. The project includes curriculum development and incorporation of teacher, teacher aid, classroom volunteer, and parent training. Early childhood language and reading readiness curriculum is presented to teach the following computer skills: 1. Learning letters and numbers, locating keys on a keyboard on the floor (gross motor); 2. Pressing correct keys on a keyboard; 3. Learning special function keys; 4. Flowcharting (understanding shapes with meaning stories, traveling the giant flowchart-on-the-floor path, copying, and creating flowcharts); 5. Learning directions and programming in LOGO and SAVVY; 6. Commanding a robot using Robot Control Language and SAVVY; 7. Use of hardware and software; and 8. Software enhancement of early childhood developmental curriculum.

A brief summary of results will be discussed; however, the majority of the program will be demonstration of activities and materials. Consumables are available.

ABSTRACT: Effective Technologies with Young or Handicapped Learners: Touchscreens and the Microcomputer

Pamela Campbell, Center for the Study of Social Acceptance, University of Massachusetts, Downtown Center, Boston, MA 02125.

Matching-to-sample is an instructional procedure that has proven effective across many types of learners, settings, and instructional tasks. In this procedure, a sample stimulus is presented to the learner who then selects and matches a response from alternative choices surrounding the stimulus. Matching-to-sample has demonstrated that learners can gain new information quickly and accurately and retain these gains over time.

Microcomputers that use touchscreens for responding make it possible to implement precise, efficient matching-to-sample programs that are particularly appropriate for very young or handicapped learners. Touchscreen and microcomputer technology offer effective, efficient, appropriate, and reliable means with which to learn new information.

A recent investigation will be described in which the number of choices presented to the learner was studied with preschool-aged learners. Three- and six-choice response choice conditions were manipulated to determine the effects upon accuracy, speed, learning, response choice position, and maintenance. Findings have clear implications for the design of effective instruction (set size, reinforcers, attention, time-on-task), the use of appropriate technologies, and the interchanging of research and teaching roles.

ABSTRACT: Enrichment vs. Remediation for At-Risk Students

Antonia Stone, Executive Director, Tom Harris, Arthur Wood, Playing To Win, Inc., 1449 Lexington Avenue, New York, NY 10128.

Imagine the basement of a housing project building in East Harlem (NYC), pipes on the ceiling, asphalt tile on the floor. In between are thirty or more computers of six different makes. Every afternoon, Monday through Friday, at 3:45, the doors of this basement space are thrown open to any young person from the area who wants to use a computer. By 3:15, the line is long. Those up front will have the first choice among the software programs available, so many youngsters dash over as soon as school is over.

This program is a project of Playing To Win, part of the activities of the Playing To Win Computer Center. It is part of Playing To Win's effort to enrich the lives of the children of East Harlem and their families through access to technology. Although the term "at-risk" is subject to interpretation, the residents of East Harlem qualify under almost any definition. Half of the area children will drop out of high school, close to 50% of the families receive public assistance, and the unemployment rate is double the average rate for NYC, already one of the highest in the country.

Many of the children who come to the Center have had little experience of computers in school. When they have, it has been drill and practice offered through their remediation programs in Special Education or Chapter I classes. By giving these children the opportunity to use computers for what they want, to ask questions when they need answers, to work at their own pace toward their own goals, Playing To Win seeks to provide the enrichment from which they have been "programmed out" in school.

The Center has been operating in East Harlem for close to six years. Just this past September, Playing To Win opened a second center in Somerville, Massachusetts. Tom Harris, supervisor of the East Harlem enter, and Arthur Wood, supervisor of the Somerville Center, together with Antonia Stone, Executive Director of Playing To Win, will address the dichotomy between enrichment and remediation from the point of view of their experience and will illustrate their remarks with a videotape of activities at the Playing To Win Centers. Time will be reserved for questions from attendees.

INSERVICE TRAINING: A COOPERATIVE APPROACH

Sylvia Charp, Chair
T.H.E. Journal
2626 S. Pullman, Suite 250
Santa Ana, CA 92705

ABSTRACT

It is widely recognized that inadequate training for teachers and staff is often an obstacle to implementing computers in education. This session will be a panel discussion among several leading computer and software vendors on this subject. Each panel member will give a short presentation, followed by a general discussion. Subjects that will be addressed include: identifying the inservice needs of the educational community as they relate to computer technology and curriculum integration; methods for addressing those needs; models of success; and ways to promote an industry-education partnership to solve these problems. Emphasis will be on what vendors should and could be doing better to help schools provide ongoing inservice training in computers and software. The discussion will then be opened up to the audience to encourage an exchange of information between educators and vendors.

PANELISTS

Dave Brittain
Florida State Department of Education
Talahasee, FL

Sandra Hogan
Aldus Corporation
Seattle, WA

Kenneth Rossano
Bank of Boston
Boston, MA

Frank Withrow
U.S. Office of Education
Washington, D.C.

TEACHERS TEACH AS THEY ARE TAUGHT: TEACHING LOGO TO TEACHERS

Nancy Cunniff
Center for Intelligent Tools in Education
Box 8
Teachers College * Columbia University
New York, NY 10027

Abstract

This paper discusses one approach to educational computing in teacher education. Given that we know that even today many Teachers *Teach as They Were Taught*, the paper describes a approach used to teach Logo in an environment where teachers-as-students are not only learning Logo as a programming language, but are learning to think about the philosophy, educational implications, strategies and materials for teaching Logo. The paper stresses that teachers should be encouraged to develop a philosophy of education that encompasses the teaching and use of Logo not solely as a programming language, but as a medium of expression for creative and problem solving activities. A graduate level course for educators designed to embody this approach is described. While the paper specifically discusses the teaching of Logo, the approach proposed can be applied to the teaching of any technology-related topics to teachers.

Introduction

> The point is that Logo is not a person; it neither promises not delivers. It's a medium of expression, for both children and teachers; and when I travel across the country and the world seeing Logo in many different contexts and in many different classrooms, I am struck by the variety of forms it takes in these different settings with different teachers and different children. (Papert, 1985, p. 3)

The adage that teachers teach as they were taught proves itself to be true over and over again. Excellent teachers often attribute their success to the influence of a former instructor, professor or a favorite elementary school teacher. There are also, unfortunately, many examples of rigidity in approach and style, adherence to traditional exercises and teaching techniques, and failure of innovations due, in large part, to the influence of past experience and models. Enter computing into the classroom: how does the adage hold? While much of the difficulty in effecting educational change in traditional curriculum areas lies in the strength of tradition and the resistance to doing things differently, the same is not true of computing. One of the golden opportunities open to those of us who are involved in educational computing is that it does NOT have a history or a tradition. How can teachers "teach as they were taught" when it comes to an essentially new subject area? How can teachers "teach as they were taught" using a technology that did not even exist when they were students?

In training teachers to use technology, there is a rare opportunity for teacher educators to actually form the way teachers are taught and to influence what goes on in the nation's classrooms. The strategies, materials and processes required for successful, effective use of computing significantly differ from those that are part of the average teacher's repertoire. Although there are always a handful of gifted teachers who, after careful consideration of any new technology or material, can intuitively devise creative strategies and curriculum integration plans, most teachers need carefully devised pre-service and in-service education to help them develop competence in using and teaching with those technologies. Teachers will teach as they are taught, therefore, teacher training in technology must incorporate models of how technology should be used in schools.

This paper is a discussion of an approach to teaching Logo. It describes a one-semester graduate level course for teachers. It suggests that one of the reasons that the teaching and use of Logo has not "taken hold" in schools is that the way we have gone about training teachers to use Logo is not sufficient. The paper describes an approach that integrates the learning of Logo with the development of a philosophy, teaching strategies and materials for using computing in general and Logo in particular. While the specific content area discussed herein is Logo, the model for teacher training suggested is generalizable to other technology-based content areas.

Training Teachers to Use Technology:
Meeting Reality Head-on

Two of the most commonly used approaches to teacher training for using technology are:

1. *Teach CONTENT: train teachers to know about technology.* Using this approach, teachers are taught how to use a word processor, a database, CAI software or how to program in Logo, BASIC or Pascal. The training, often because of time limitations, only focuses on content, and while the teachers-as-students learn about an application, they are left to devise for themselves ways of using their new knowledge.

2. *Teach METHOD: train teachers to use technology in classrooms.* This approach focuses from the start on using technology in the classroom. It assumes that the teachers-as-students will somehow learn the content involved. Curriculum and integration issues, strategies and materials are discussed.

It is easy to understand why teacher training programs feel the need to focus on one or the other of the above approaches. Most training programs are short -- a couple of hours on an in-service day, a weekend day or a few days during the summer. While these efforts should not be criticized too severely, it is imperative that more effective models be proposed, tested and institutionalized so that we can begin to train technology-using educators more systematically.

One of the most effective ways of combining content and method is to present content in a way that models appropriate methodology and uses materials and strategies appropriate for use in classrooms. Since most teachers have had little experience *using* technology, *learning with* technology or *teaching with or about* technology, such a modeling approach can be particularly effective. When teachers are trained by those who provide good models, they can then effectively "teach as they were taught."

Of course, such an approach requires a significant time commitment: learning about computers takes more than a short term workshop; learning how to teach about or teach with computing takes more time than is available in a brief in-service course. Teachers-as-students need to learn about computing -- they must become "owners of the technology" insofar as they must be comfortable using it. They also need to develop strategies, materials, and organizational plans. Most importantly, they need to consolidate a philosophy of education that includes the use of technology as an integral aspect of the learning environment.

A Logo Course for Teachers

The course that will be described here is entitled *Teaching Computing to Children Through Logo* and is a three semester-hour course at Teachers College, Columbia University. Teachers College is a graduate school of education, and many of our students are practicing teachers who come to study educational computing, many of whom have little computing experience. The range of Logo knowledge and experience and knowledge is broad. Occasionally, a student enrolls who knows a great deal of Logo; usually, several students have a passing acquaintance with Logo -- they have read an article or two, have attended a half-day or day-long workshop. Many, though, have no experience, and were directed to the course by enthusiastic advisors or peers.

Rationale

Most adult students of Logo are teachers or would-be teachers who are learning Logo for the purpose of teaching it to their own students. Many of these teachers-as-students have little or no computing experience, yet expect (or are expected by their administrators) to master enough computing in short-term workshops to be able to use it in their classrooms. Those workshops usually follow one of the two patterns described above, either exclusively focusing on the technical aspects of the language itself or on the teaching of Logo. Rarely is there time for both. If Logo will ever be used effectively by more than a handful of creative, committed teachers, courses and workshops for teaching Logo to teachers must become more comprehensive. Models must be designed and tested.

Course Description

The course description given to students includes the following comments:

> This course is both a course in programming in LOGO and a course in teaching computing to children using LOGO. The course is, essentially, a combination of lecture and lab, and is designed to teach you how to teach and use LOGO with children by providing you with an experience of being the learner. Your reflections on your own learning experience will be most valuable as you begin to contemplate using LOGO with children. It is important for you to realize that in order to think about teaching LOGO, you must be quite knowledgeable about it. It is the focus of this course to help you begin to explore the LOGO language and the ramifications of using computing and particularly LOGO in education.
>
> One of the vital elements of programming in LOGO is the establishing of what is called a "LOGO ENVIRONMENT." Part of that environment requires programmers to work together as partners. This environment requires that all of us work together actively, discussing problems and successes. It calls for a sharing of ideas and programming tasks. Some of the work we will do in this class will require that you work with other students. This provides a built-in resource for discussion, and it can lead to a great deal of productive thinking and programming.
>
> Finally, one of the nicest things about programming in general, and especially about programming in LOGO, is that it encourages you to think about your thinking. You have to spend time grappling with what you will at first call "mistakes," but which you will eventually learn to call "bugs" -- and maybe will come to think of as opportunities. In programming, you often learn that there are very many paths to a solution, not just one "right" path.

While the course title might suggest that the content focus will be the discussion of teaching strategies, curriculum design and/or examination of research, the students are informed at the first class that they cannot begin to think about the significant educational issues surrounding Logo if they do not know Logo themselves -- if they have not experienced a "Logo Environment." The course, then, has two main strands that are interleaved continuously: the students learn the Logo language and the students learn to think about the educational uses of Logo.

Following the college schedule, *Teaching Computing to Children* is a one-semester course that meets once each week for fifteen weeks. Each three hour class session is a combination of directed teaching, discussion, and individual and small group lab work. Periodically part of the class is devoted to a group project involving the whole class. The course is taught in a microcomputer lab of IBM-PCjr computers. The room is equipped with printers, a large display monitor and an overhead projector. We have access to a lab of Apple IIE computers; one class a term is generally taught in that lab so that the students have the experience of using Logo on a different kind of computer.

We are use LCSI's *LogoWriter*; each student is given a copy of Logo for completion of his/her work during the course. The ability to give students the software is a critical factor in the success of the course. Since most of our students are graduate students enrolled in a degree program, many have access to some type of computer at home; since most are teachers, many have access to computers at work. Because we can give them the software, they are able to work on their Logo projects without having to come to the College's computer labs. For part-time students, this is a positive aspect of the course. In fact, it helps to overcome one of the most common difficulties of computing courses: access to software and hardware.

What Do We Teach?

Training teachers to use Logo creatively and in a way that is pedagogically sound requires a carefully considered educational plan. The course of study for teaching teachers to use Logo must have three-pronged focus: learning Logo as a programming language, learning to use Logo as a medium of expression (not simply as a programming language) and examining strategies, materials and philosophies for teaching and implementation in classrooms.

Mastery of the concepts and structures of Logo as a language is primary and essential; without that, teachers cannot be expected to teach Logo or to support students in the use of Logo for curriculum projects. However, while teachers-as-students are learning Logo as a language, the teaching-learning environment should reflect the fact that Logo is more than a programming language: it is, in fact, the foundation and cornerstone in an environment where it is used for solving problems and creative expression.

In the teaching of Logo to teachers, there should be a distinction between Logo as a language and Logo as a medium of expression. Just as students cannot use music, art or writing as media for expression without knowing the basic concepts and fundamental structures of those disciplines, so students cannot use Logo as a medium without mastery of the fundamental structures and concepts of the language itself. So, too, must teachers who will use Logo with students have a mastery of those fundamentals. And so, students of Logo, be they elementary school students or adult learners, should be taught the fundamental concepts and structures of the language. It is only after those foundational concepts and structures are mastered that real problem solving and thinking can take place. Teachers-as-students should be taught in a manner that reflects the best of what we know about creative, problem solving environments. Discussions of teaching strategies and materials, organizational plans and classroom management must be explicit.

How Do We Teach?

One of the most widely used strategies in teaching Logo is the "Project Approach" in which students learning Logo engage in a long-term programming project, usually in collaboration with one or more peers. The project approach is seen as discovery learning in which student programmers explore the language and engage in creative problem solving. Such an exploratory, discovery-based learning experience is thought to be "true" to the spirit of Logo, and projects are often seen as "much, if not most, of what a Logo environment experience should be" (McNerney, 1988, p. 36). Because this approach is seen as so important to the "spirit of Logo," the project approach is often used almost exclusively in Logo courses for teachers. After brief introductions, students begin working on a single, long-term project. The instructor helps to focus the students, getting them started, encouraging and watching out for teachable moments.

While the project approach has many positive aspects, using it as the sole backbone of an introductory workshop or course for teachers has severe limitations. Primarily, teachers who come to study Logo are doing so for a very specific purpose: to learn enough Logo and enough about teaching Logo so that they can successfully use Logo in classrooms. They are not simply students studying Logo for a "pure" educational experience. The project approach, while often enjoyable and rewarding as a learning experience, is often rejected by teachers-as-students as too narrow. They feel the need to explicitly define the scope of what they should be learning and they are anxious to deal with the content and structure of the language and the pedagogical aspects of it "head on." In many cases, teachers-as-students would opt for a course that incorporates more directed teaching, discussion of pedagogical issues and illustration and experience with classroom-based ideas. The project approach leaves them feeling that they are being "left alone" to figure out what they should be learning.

The challenge, then, is to find a compromise between how to "teach teachers as we wish them to teach" and how to "teach teachers as they wish to be taught!" The compromise discussed here is a modified approach: one that combines directed teaching with discussion and a variety of planned and discovery-based short and long term projects. One of the underlying goals of the project approach is to model a way of teaching for these teachers-as-students. It should be a component of a Logo course, not the sole focus.

There is disagreement about the amount of "direct teaching" of Logo that should take place, either in a class for adults or children. Heller, Martin and Wright (1985, p. 3) discuss three models for teaching Logo to children:

- -completely open-ended, exploratory and learner-directed
- -completely directed, based on a set of "exercises" that focus on a pre-determined scope and sequence of Logo skills to be mastered
- -guided discovery, a middle ground where students are taught Logo skills and are then encouraged to explore them in self-directed projects.

The same three models are used in Logo workshops and courses for teachers. In many cases, the first model, the open-ended, exploratory approach, does not work with teachers. This is not because, as many suggest, that teachers cannot be exploratory learners. Rather, it is because the teachers-as-students realize how limited their learning time is. While they may or may not understand the strategy behind the open-ended workshop approach, they do know that open-ended learning is time intensive. They want to be taught, to get as much information and knowledge out of a short term experience as possible.

On the other hand, the second model, what I call the "exercise approach," is counter to the idea that we should teach they way we want our teachers-as-students to teach. When we present an "Exercise Book" version of Logo, we are implicitly telling teachers to use that approach in their own teaching of Logo. We are giving them tacit approval to teach Logo as another "subject" rather than to create an environment where Logo is used as a medium for expression and problem solving. Thus, the best design for a Logo course for teachers combines the models; some directed teaching of Logo constructs and concepts, discussion of application, exploratory programming and reflection on the process.

Teaching Logo Language Constructs and Concepts. In this course, the major concepts and structures of the Logo language are taught directly. They are presented using a wide variety of strategies designed to illustrate effective ways to introduce concepts to children. Materials are demonstrated and discussed. Two examples are illustrated below.

1) Recursion. In the presentation of recursion, several different kinds of explanations are given, each using different materials, one of which is illustrated below. Several copies of a procedure are used, each with its own "memory jar." The recursive procedure is executed by students, each "walking through one copy" of the procedure, filling each "jar" with the appropriate value, and checking off lines of the procedure as they are executed. It is not until each student executes the END of his or her copy of the procedure that the recursion is completed. The act of "executing" each line of the procedure by acting it out helps students build a understanding of the concept of recursion. (See Fig. 1.)

2) A simple picture-filled grid is a material used to introduce many Logo concepts from FORWARD and BACK to SETPOS and SETHEADING. The use of the same materials for discussion of multiple concepts helps students make connections between and among the basic ideas of the Logo language. Using a small plastic turtle, the grid can be used to develop the concepts of MOVING and TURNING the turtle, setting the turtle's HEADING, and setting the turtle's POSITION. (See Fig. 2.)

Activities and Projects. The key element that keeps this course from dissolving into simply a programming course is the focus on the development of projects and activities that are pedagogically sound, that support the goals of the curriculum of a specific grade level or subject area, and that require the students (be they the teachers in this course or the children in an elementary classroom) to continually expand and *use* their knowledge of Logo and problem solving. During the semester, several different kinds of projects are executed during class lab periods, two of which are discussed below. All students are also required to complete several curriculum-base mini-projects and one longer, more extensive, traditional "Logo Project." The use of many different kinds of projects helps teachers-as-students realize the expansive potential of Logo,not only as a language, but as a problem solving tools, and most especially as a medium of expression.

One Computer -- Whole Class: The Progressive Picture. One concern continually voiced by teachers is that it is difficult, if not impossible, to use Logo effectively in a classroom with only one or two computers. The goal of the following project is to give teachers-as-students an experience in the planning and development of a group project that can be used effectively with large groups of students using one or two computers. The project is described to students as follows:

> This project is designed to allow you to work with a group of children and one (or a small number of) computers. It is meant to
>
> - -introduce children to the idea of sub- and super-procedures,
> - -give them an experience in group problem solving,
> - -create a positive problem solving environment where cooperation is the key to success,
> - -result in a project that is "larger" and more involved than most could do alone, and
> - -provide an environment for working on a "long-term" project.
>
> The project has several phases, each one being a crucial component and necessary for the success of the project as a whole. Briefly the stages are:

1. *Introduction by the teacher*: discuss progressive stories and progressive pictures. As a class, choose a topic for a progressive picture. The topic can correlate with a social studies or science topic being studied. The class "pairs off."

2. *Discussion and Drawing*: Focusing on the topic chosen, set the ground rules. Using a large sheet of drawing paper taped to the wall and a set of colored markers, have each pair or individual add ONE ITEM to the drawing.

3. *Setting Up Planning Charts*: Discuss the fact that everyone must stick fairly closely to the plan. Make a place for public note-taking -- keeping track of turtles, colors and coordinates.

4. *Planning and Programming*: Each pair of students plans and programs their piece of the project. This can take place over a couple of weeks or more if there is only one computer in the room. Daily printouts of the procedures and the graphic output should be made and posted.

5. *Gluing Together*: LOGOWriter makes this particularly easy. If the actual programming is done on more than one computer (or disk), use the command LOAD "pagename. The procedures on the flip side of "pagename are loaded onto the flip side of the current page.

6. *Debugging and Running*: Once all of the pieces of the picture have been programmed, the group should convene for a group debugging session. The primary purpose of this session with teachers-as-students is that it allows you to model good debugging strategies. This is also the time to write a "superprocedure" and talk about modularization of big projects.

GROUND RULES

-All procedures should assume total responsibility for the positioning and heading of the turtle. Do not assume that the turtle will be at HOME when your procedure is called.

-All procedures are responsible for TELLING the turtle they are using and setting up the correct state for the turtle (e.g. PU or PD, hiding or showing the turtle, etc.)

-The group decides collectively on the BACKGROUND COLOR.

-If you are making a new shape, add it to the group list -- so no one else will use the same number.

-Remember, there are only 4 turtles. Stick with the original plan and don't try to use more animation than is possible.

See Fugure 3 for an example of the graphic output of a group "Progressive Picture" created during one two-hour lab period at the beginning of the semester.

"Online Books" Project. This second project is designed to illustrate successful small group work. The concept of this project is that it can be worked on over a long period of time and that each group member, although essentially programming alone, has a responsibility to the larger group to follow guidelines and complete his/her task within the parameters set forth by the group. It is an ideal way to teach about LOGOWriter's workspace management. The project is described:

> This project is designed to be executed by small groups of students, and is ideally suited for classes where there are a limited number of computers. Ideally, groups should be between four and six members, but this project idea can also be implemented by a whole class of as many as 20 to 25 students (of course, groups of that size would take a significantly longer time to complete the project).
>
> Write a program that simulates a "Picture Book." Just as books have many pages, your completed program will have many pages -- one page to be written by each member of the group. The first page will be the book's title page, and should give all appropriate credits. It will then call upon the book's opening page. Each successive page will present one picture or concept and will then get the next page. The final result will be a little like flipping pages in a picture book -- you get to look at one page, then another one appears.
>
> Briefly, the stages of this project are:
>
> 1. As a group, decide on a theme for your book and assign one page to each group member. Coordinate book themes with topics being studied in other classes: illustrate different perspectives, shapes, or relationships being studied in mathematics; create a picture dictionary to be used for foreign language students; illustrate homes; create an alphabet book.
>
> 2. Individually, each group member creates a page. Each page should have a superprocedure that drives the page and a procedure called STARTUP which should:
>
>   -clear text, graphics and commands
>   -invoke the superprocedure that runs the page
>   -call the tool procedure WAITFORKEY (which waits for the user to press any key)
>   -GETPAGE "nextpage -- where NEXTPAGE is the name of the page written by the person whose page logically follows yours.
>
> 3. Put all of the pages on one Scrapbook disk.

4. Run the program and debug it.

While the "Book Project" is designed to be executed during one class session, discussion that follows focuses on the idea that this may be a long-term project in a classroom. We discuss many different curriculum-related ideas for online book projects. The project requires a significant amount of discussion before the work begins, and should be followed by a group sharing of the "completed books." In this class, it is not the sophistication of the project pages themselves that matters, but the development of the idea of individuals working within a group to produce a cohesive whole. The students are encourage to keep their pages extremely simple (in fact, the severe time limitation imposed dictates this) to illustrate that even very simple pieces can add up to a whole that is greater than the sum of the parts. (See Fig. 4.)

Final Projects. During the second half of the semester, students work on a traditional "Logo Project." Some lab time is devoted to that work, but since the project is one of the major course assignments, students are expected to work outside of class time. Since students are encouraged to select themes of personal interest for their final projects, the topics are vast, ranging from:

-Plants and how they grow
-Holidays around the world
-Vacation spots
-Mapping: An Introduction to Different Projections
-Create a Greeting Card
-The Tortoise and the Hare: An Illustrated Story
-Interactive Adventure Game

These projects are far too extensive to illustrate here; all require multiple pages in LogoWriter. Students are encouraged NOT to focus on developing computer-assisted instructional programs for the final project, but to develop a program that explores and illustrates a concept, develops a microworld or allows for creative expression in an area of personal interest. The idea here is that Logo should not be seen as a vehicle for the development of traditional CAI, and it is too easy for teachers to fall into the "CAI mode" when thinking about computer applications.

OUTCOMES: Summary and Recommendations

After participating in this course, what happens to the teachers? What happens to their students? News from the field indicates that the kinds of projects discussed and experienced during the course are used, expanded and enriched in classrooms. One former student is a primary grade teacher. His class has created "Online Books" about several science and health units. Another teaches Logo in a lab setting. His junior high school students are working on long-term projects such as a Logo Monopoly game. Many of the students in the class have reported that their own mastery of the language has been critically important in helping their own students. However, they have also reported that the wide range of assignments and projects required has helped them to be able to teach and use Logo more creatively with their own students.

This paper describes an approach to teacher training used in a semester-long course for teachers and would-be teachers of Logo. The course has a dual focus: the teaching and learning of the Logo language itself, and the development of excellence in the teaching and use of Logo in the classroom. The enthusiasm for the teaching of Logo and the use of Logo in a problem solving environment has been evident. Many students begin teaching Logo before the course is completed; many have started since then. The enthusiasm for Logo and its educational implementation has been encouraging.

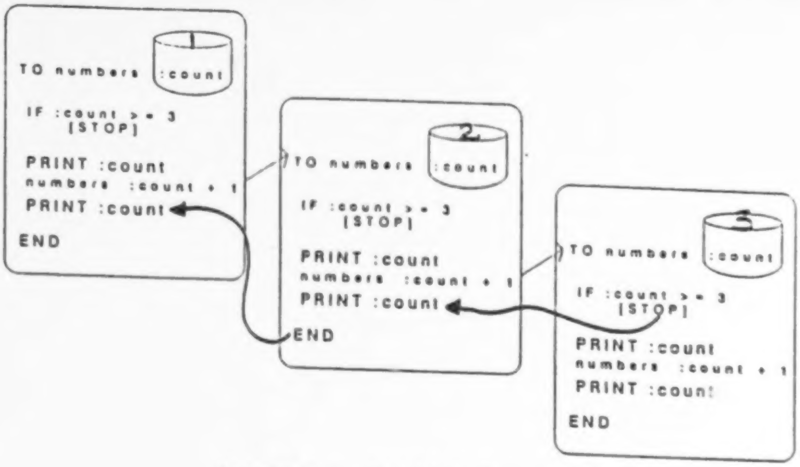
The projects completed during the course have been of a level that suggests an excellent mastery of Logo as a language. More importantly, the students have experienced a wide range of project configurations and have "lived with" different approaches to configuring computing in classrooms. The course labs, designed to illustrate differing patterns of computing in classrooms and labs provides opportunities to discuss the pros and cons of each.

Group problem solving and debugging sessions model strategies that promote sharing, cooperation and discussion. Different grouping strategies (pairs, teams of four or five, groups of six to eight and whole class) for projects underscores potential uses in a variety of classroom environments. Thus, the second goal of the course, to help teachers become excellent teachers of Logo, is advanced by means of creatively modeling the kinds of teaching that can be implemented in an elementary or junior high school classroom.

The course described in this paper is certainly not the only model of successful teacher training. It is successful because it provides the teachers-as-students with a firm foundation in the Logo language while at the same time providing an environment where teaching and learning are discussed in context.

References
and
Suggested Readings

Heller, R.S., Martin, C.D. & Wright, J.L. (1985). *Logoworlds.* Rockville, MD: Computer Science Press.
McNerney, C.R. (1988). The value of a group project. *Logo Exchange*, 7:1, September, 35-36.
Papert, S. (1985). Different visions of Logo. *Computers in the Schools*, 2:2-3 (Summer/Fall), 3-8.
Papert, S. (1980). *Mindstorms.* NY: Basic Books.
Snyder, T. (1988). Tools for teachers. *The Computing Teacher*. 15:1, 8-16.
Tipps, S. & Bull, G. (1985). Teachers need time for turtles: Planning for teacher development with Logo. *Computers in the Schools*, 2:2-3 (Summer/Fall), 273-281.



```
TO numbers :count
IF :count >= 3
    [STOP]
PRINT :count
numbers :count + 1
PRINT :count
END
```

FIGURE 1: Teaching Recursion

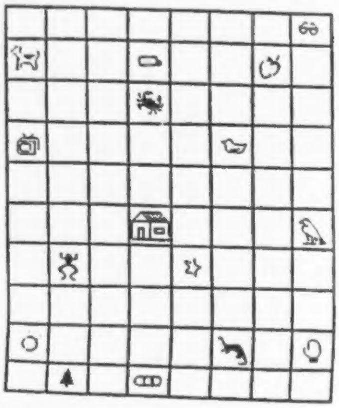
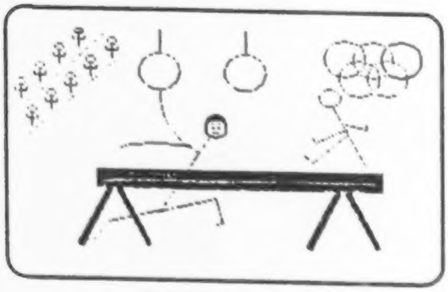
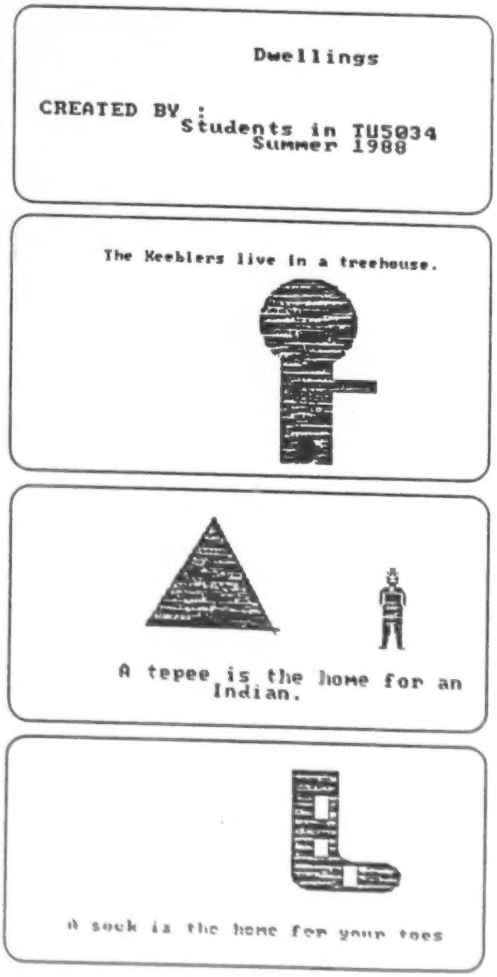


FIGURE 2: Grid for Teaching Basic Commands



AT THE OLYMPICS

FIGURE 3: A Progressive Picture



Dwellings

CREATED BY :
Students in TU5034
Summer 1988

The Keeblers live in a treehouse.

A tepee is the home for an Indian.

A sock is the home for your toes

FIGURE 4: Online Storybook

THE ELEMENTARY MATH AND TECHNOLOGY PROJECT: A MODEL FOR TEACHER ENHANCEMENT

Ann Grady
Office of Technology
Boston Public Schools
75 New Dudley Street
Boston, Ma. 02119

Abstract

For the past year, the Boston Public Schools has conducted a Teacher Enhancement Project, under the sponsorship of the National Science Foundation. The goal of the project is to develop a core group of Mentor Teachers who work to integrate technology into their own elementary mathematics instruction. These Mentor Teachers work with teams of teachers, new to teaching. Over a three year period, at least one teacher from each of the seventy eight elementary schools in Boston will participate in the project. New participants, in turn conduct workshops and act as mentors for the other teachers in their schools.

The Elementary Math and Technology Project (EMAT) was funded in May 1988 for three and a half years. Eighteen teachers from as many schools were selected by competitive screening as Mentor Teachers. These teachers participated in a Summer Institute to enhance their own mathematics backgrounds, review software, and develop curriculum units which integrated manipulatives and technology. During the 1988-89 school year, they piloted materials with their own classes and explored effective use of computers and telecommunications in their classrooms.

Already, the "EMAT Model" has had a significant impact on the individual teachers involved in the projects, as well as on their colleagues and schools. Teachers and principals report a renewed energy for professional development, increased reflection on teaching, and enhanced collaboration among teachers in one school and between schools. The "EMAT Model" seems to empower teachers and to increase significantly, the impact of professional development.

The impact of the EMAT grant is being felt earlier and more extensively than predicted when the proposal was written. We anticipated that fertile ground existed in the schools, ready to be nurtured by a project which combined technology, the excellent resources of area university mathematics faculty, and our own best teachers. The mixture seems to be working far beyond our expectations.

Introduction

For the past year the Boston Public Schools has conducted a Teacher Enhancement Project under the sponsorship of the National Science Foundation. The goal of the project is to develop a core group of Mentor Teachers who work to integrate technology into their own elementary mathematics instruction. These Mentor Teachers work with teams of teachers, new to technology, as they begin to incorporate technology into their teaching. Over a three year period, at least one teacher from each of the seventy eight elementary schools in Boston will participate in the project. New participants, in turn conduct workshops and act as mentors for the other teachers in their schools.

Background

Boston Public Schools has been a leader among large, urban school districts in the development of a system-wide instructional technology program. By the fall of 1987, a minimum of one computer lab was in place in every school in the city, and a K - 12 curriculum consisting of applications (word processing, data base, spreadsheet); programming (Logo, BASIC, Pascal); and computer literacy was required of all students. Substantial numbers of teachers voluntarily enrolled in professional development activities (graduate courses and workshops), jointly sponsored by the school system and local universities. Individual teachers had begun to use computers in their classrooms to enhance the regular curriculum. These teachers spoke enthusiastically of the impact of the computer on their own teaching, as well as, on student learning.

At the same time, the school system conducted an extensive self-evaluation in all subject areas (the Boston Education Plan). The Mathematics Task Force identified professional development in mathematics for elementary teachers as a high priority. The Task Force also recommended teacher training in the use of manipulatives, calculators and computers.

In 1987, Boston Public Schools submitted a proposal to the National Science Foundation Division of Teacher Enhancement to develop a core group of elementary teachers, who would learn to use technology with their own classes and subsequently instruct other teachers. The proposal was built on the premise that there are excellent elementary teachers who are willing and excited to explore the use of technology in their instruction. The current structure of schools makes it difficult for these teachers to share their knowledge with fellow staff members. We proposed releasing these teachers periodically to meet with one another, to explore new uses of technology, and to team teach with teachers new to the project.

In addition, the Boston area has excellent resources in both mathematics and technology education. Dr. Robert Willcutt of Boston University, nationally recognized for his curriculum development and inservice work with teachers in manipulatives and problem solving, agreed to serve as our Principal Mathematics Consultant. National Science Foundation Curriculum Development Projects based in Boston at The Educational Development Corporation (Reckoning with Numbers), and the Technical Education Research Center (Used Numbers), as well as Seymour Papert's Project Headlight agreed to collaborate.

The Elementary Math and Technology Project (EMAT) was funded in Spring 1988 for three and one half years. A Planning Committee composed of the Directors of Math and Technology for Boston Schools, university representatives, and reachers developed selection criteria for the participants, planned the content of the Summer Institute, selected consultants, and reviewed pilot math/technology curricula for possible inclusion in the project.

Teacher Selection

Notice of the project was posted at all elementary schools in the city. Teachers responded enthusiastically and applied in great numbers to participate in the project. Each applicant was visited in his/her classroom by a member of the screening committee. Each teacher's principal was also interviewed concerning the teacher's leadership role in the school as well as his/her teaching effectiveness. The two major criteria used in teacher selection were demonstrated excellence as a classroom teacher, and an interest in and commitment to improving mathematics instruction through the use of technology.

Eighteen teachers from as many schools were selected as Mentor Teachers. Since some schools had several strong applicants, additional teachers from these schools were identified as members of school teams. Project participants' computer backgrounds ranged from expert to novice, with most teachers having some experience in the use of educational software.

Summer Institute

Project Participants attended a three week Summer Institute conducted at the Hennigan School, Boston's Technology School of the Future. Dr. Willcutt immediately engaged teachers and visitors alike in "stretching their minds'. Math manipulatives took on new meaning as he presented one exciting new math idea after another. Everyone worked as a team to solve interesting and fun problems, while discussing what should be taught according to the new National Council of Teachers of Mathematics standards (more problem solving, estimation, and communication in math).

Kindergarten through grade 5 teachers in all areas (Bilingual, Special Education, Regular Education, Chapter I, Advanced Work, and Computer), shared ideas that worked with their classes. Teachers made presentations on LogoWriter, Lego Logo, and the Geometric Pre-Supposer. Publishers loaned more than 100 pieces of software to the Summer Institute for review. Team members worked to identify software which enhanced the teaching of math concepts. Teams developed lessons which incorporated manipulatives and computers, as well as other activities for those "difficult to teach" topics in elementary mathematics.

During the Summer Institute, EMAT teachers were divided into teams by grade level, with each team led by a Team Leader. Team Leaders were exemplary classroom teachers chosen for their extensive backgrounds in mathematics and educational technology, and for their recognized leadership skills. The teams developed lessons which incorporated manipulatives, computer exercises, and other written or oral activities. These lessons will teach some of the most difficult to teach topics of elementary mathematics. Teachers are piloting the lessons and materials during the 1988-89 school year, and revising them for distribution to all Boston teachers during the workshops they will conduct in succeeding years.

Teachers also learned to use telecommunications during the Summer Institute. The project provides teachers with modems and phone lines so teachers and students can communicate mathematically from classroom to classroom, and challenge one another with the "Problem of the Week". Teachers left the Institute eager to implement their newly acquired ideas and technology in classes in September.

Summer Program Evaluation

An evaluation conducted by an independent evaluator, showed almost unanimous satisfaction with the Institute. Teachers commented especially on the opportunity to work with one another and with Bob Willcutt and other consultants, to look at mathematics and the teaching of mathematics from a fresh perspective, and to discover together, a new and effective ways of incorporating technology into their teaching. An excerpt from the evaluation is included in the Appendix.

1988-89 School Year Implementation

The month of September was filled with ordering and delivering equipment and supplies, and installing phone lines. Each teacher received two computers for her/his class, a modem and a budget for software and manipulative materials.

Funding for substitutes was provided so Mentor Teachers could be released once a month to continue the work of the Summer Institute. Teachers arrived each month eager to share new teaching ideas, successes and struggles with one another. Observers noted a perceptible level of energy and enthusiasm for mathematics within the group. Teachers reported the struggles required to prepare for the release day; one teacher even recruited a parent to teach her class at the last minute (no easy task in an urban school system) when the assigned substitute did not appear.

Dr. Willcutt continued his mathematics content presentations in each session, including additional topics at teachers' requests. Teachers commented that these content presentations provided them with an opportunity to view the elementary mathematics curriculum as a continuum; kindergarten teachers saw how the concepts which they introduced were expanded and refined in the upper grades and fourth and fifth grade teachers could share their concerns for preparation of students at lower levels.

Teachers expanded significantly, their use of manipulatives for the development of "difficult to teach" mathematics concepts. They also developed problem solving approaches to the teaching of basic skills operations and reported that students developed a more solid understanding of concepts in a shorter period of time using manipulatives and the computer than with the standard textbook and worksheet presentations. Some teachers reported that their students asked to use the computer rather than go outside for recess.

Teams met to reflect on their experience piloting materials developed during the summer and to make revisions based on that piloting. Team members shared ideas, software, and manipulatives with one another. During one session, fourth and fifth grade teachers brought their best materials, games, and software for teaching fractions. More than twenty different exciting ideas were exchanged and teachers formed a writing group to develop a cohesive unit which utilized the best ideas shared by the teachers.

Between monthly meetings, Dr. Willcutt spent one day each week visiting teachers and classes, working with teachers to refine mathematics concepts and presentations developed during the monthly meetings. Teachers and students alike waited enthusiastically for his visits. One group was out at recess when he arrived, and called out to one another, "the professor is coming, the professor is coming!!" Teachers worked with Dr. Willcutt to adapt the materials for all ages and levels, from learning disabled students to the gifted and talented.

The Project Director and Coordinator also visited classes and worked with students and teachers. These meetings provided an opportunity to develop classroom management strategies for utilization of two computers and a variety of manipulatives with classes of twenty five or more students. The Project Staff experienced the student enthusiasm for math as students eagerly shared their expertise and projects with visitors. At several schools, other teachers have asked to share the "EMAT Project" and both teachers and students move from class to class , sharing ideas with one another.

Teachers have begun to use their release time to visit other EMAT teachers and schools. One third grade teacher had developed an exciting measurement unit with his own bilingual class. Other teachers heard about the lesson at a monthly meeting and invited him to introduce it to their classes. He adapted his materials for kindergarten, special needs, and second grade students and traveled to four teachers' classes to present the lesson. At the next monthly meeting, all five teachers reported enthusiastically of the impact of the day on them and on their students. Several teachers reported that EMAT has given them a unique opportunity to observe their own students being taught by another teacher. They feel that they have greatly enhanced their understanding of their students' learning through these observations.

Teachers use telecommunications extensively to share ideas and concerns with one another. They report that telecommunications provides a unique opportunity to "chat" with a fellow teacher in a manner more accessible than pencil and paper, and more mutually convenient than the telephone. The Project Staff also telecommunicate with EMAT teachers, sharing teaching ideas and concerns as well as routine messages.

Principals report that despite substitute hassles (substitutes are in very short supply in Boston), they are willing to release teachers one day per month because they return to school reinvigorated by their EMAT meetings. One principal, who has three teachers involved in the project, says that the day before the meeting, she has many concerns about letting her teachers attend, but the day after, their energy and enthusiasm make it all worthwhile! The teachers return with many new ideas to share with other teachers, as well as their own students.

Expansion of First Year Goals

The first year goal of the project was for teachers to pilot materials with their own classes. However, as teachers spoke to their colleagues about their experiences, their fellow teachers asked to participate. They heard about Dr. Willcutt's team teaching with Mentor Teachers in their classes and their students eager anticipation of "the Professor's" arrival. The teachers asked principals to release additional teachers to observe the lessons. Principals again reported that despite the logistical difficulties, the impact on teachers' mathematics instruction made it worthwhile.

The Boston School's Office of Technology offered Apple II+ Computers (recycled from the high schools) to elementary teachers on a competitive basis. Teachers in EMAT schools were among the first to apply. EMAT teachers helped their colleagues to identify software and reorganize their classes to use computers effectively with their classes. Clearly, EMAT teachers were being unofficially recognized as Mentors a year ahead of schedule.

EMAT teachers volunteered to pilot new technology based math materials developed by the Educational Development Corporation. They worked with suburban teachers to explore new methods and materials for teaching mathematics, emphasizing the new NCTM standards. Other EMAT teachers worked with the Technology Education Research Center to pilot the Used Numbers curriculum with their classes. Another pair of enterprising EMAT teachers wrote a grant for Massachusetts State Funds to train their fellow teachers in LegoLogo, and to purchase the necessary materials to establish a LegoLogo lending library.

Eight of the EMAT teachers were able to convince their entire faculties to sign up for after-school workshops conducted by Dr. Willcutt and other area mathematicians. These workshops will be funded by a Massachusetts State grant for teletraining. The workshops will be conducted at one school and cabled to the other sites via a one-way video and two-way audio hook up. EMAT teachers will function as teaching assistants for the course at their own schools. One hundred and fifty additional teachers will receive training this spring through the use of this new technology.

First Year Impact

Already, the "EMAT Model" has had a significant impact on the individual teachers involved in the projects, as well as on their colleagues and schools. Teachers and principals report a renewed energy for professional development, increased reflection on teaching, and enhanced collaboration among teachers in one school and between schools. The "EMAT Model" seems to empower teachers and to increase significantly, the impact of professional development.

Principals also report a renewed sense of purpose. They feel a part of "the action of EMAT" and feel a sense of pride that their school has been chosen for this special project. In some cases, whole schools have taken on a new tone, a feeling that something very special and exciting is happening in their building.

The impact of the EMAT grant is being felt earlier and more extensively than predicted when the proposal was written. We anticipated that fertile ground existed in the schools, ready to be nurtured by a project which combined technology, the excellent resources of area university mathematics faculty, and our own best teachers. The mixture seems to be working far beyond our experiences.

Second and Third Year Activities

In each of the next two years of the grant, thirty additional teachers will be chosen from schools not yet participating in EMAT. The selection criteria will be the same as during year one. New teachers will participate in a Summer Institute, and receive computers, modems, phone lines, software and manipulatives to use with their classes. Curriculum units developed by the first group of EMAT teachers will be used with their classes.

Meanwhile, the first group of EMAT teachers will plan and implement workshops for the faculties in their own buildings. They will continue the proposal writing, networking, and team building around curriculum renewal that they started the first year. Based on their enthusiasm and initiative this first year, we are certain that they will generate many additional ideas for using technology with their students. Plans are already underway to incorporate these projects into the Boston Public School's Math and Technology Curriculum.

Bibliography

Becker, H.J. (1986). *Second National Survey of Instructional uses of Microcomputers.* Baltimore, Md: Center for Social Organization of Schools, John Hopkins University.

Cowden, S.C. An analysis of an implementation model of integrating the microcomputer into elementary school instruction (Doctoral dissertation, Boston College, 1987). *Dissertation Abstracts International.*

Goodlad, J.I. *A Place Called School*, St. Louis: McGraw-Hill 1983.

Good, T.L. & Weinstein, R.S. (1986). Teacher expectations: A framework for exploring classrooms. In K.K. Zumwalt (ED.), *Improvising Teaching: 1986 ASCD Yearbook.* Alexandria, VA: Association for Supervision and Curriculum Development.

Grady, A.M. Teaching selected topics of the middle school mathematics curriculum using the microcomputer. (Doctoral Dissertation, Boston University, 1986). *Dissertation Abstracts International.*

Hatfield, L.L. (1984). Toward comprehensive instructional computing in mathematics. In V. Hansen (ED.), *Computer in Mathematics Education: 1984 Yearbook.* Reston, VA: National Council of Teachers of Mathematics.

Heath, D.H. (1986). Developing teachers, not just techniques. In K.W. Zumwalt (ED). *Improving Teaching: 1986 ASCD Yearbook.* Alexandria, VA: Association for Supervision and Curriculum Development.

Irvine, J.J. (1985). The master teacher as mentor: Role perceptions of beginning and master teachers. *Education, 106* (2), 123-130.

Lawrence, G. (1984). *Patterns of effective inservice education: a state of the art summary of research on materials and procedures for changing teacher behaviors in inservice education* (Report No. ED 176 424). Tallahassee, Florida: State Department of Education.

Lawrence Hall of Science (1986). Role of technology in the mathematics curriculum. In *Technology in the curriculum: A mathematics resource guide* (pp. 4-16). Sacramento, CA: California State Department of Education.

Lindquist, M.M.; Carpenter, T.P.; Silver, E.A.; & Matthews, W. (1983). The Third National Mathematics Assessment: Results and implications for elementary and middle schools. *Arithmetic Teacher, 31.* (4), 14-19.

Little, J.W. (1981). *School success and staff development: the role of staff development in urban desegregated schools, executive summary.* Washington, D.C.: National Institute of Education.

BOSTON PUBLIC SCHOOLS
EMAT SUMMER WORKSHOP EVALUATION

Your responses to the following questions will help the planners to understand the effectiveness of these sessions from the participants' point of view. Your suggestions will influence the future workshops and your help is appreciated.

Please comment on the following activities and rate them 1-5, (5 is the highest).

Entire Institute.

1	2	3	4	5
			2	17

Comments:

- I have taken part in many workshops in the past. I have found this experience to be one of the most educational and useful workshops I have ever attended.
- The Institute was excellent, I have not had this much fun learning and teaching others to learn, and being with other creative teachers with so much energy in a long time. This workshop has been rejuvenating, enjoyable, and a learning experience.
- This Institute was outstanding. The exposure to people, to ideas wasreally wonderful. An excitement and positive sense of anticipation has been engendered here . . . which, hopefully will continue for a long, long time.
- We could have had better ventilation. The learning process goes on though.
- This was a highly productive, totally worthwhile series of workshops.
- Best thing I was ever involved in.
- I thought that it was excellent. I was here for the three weeks. I thought that people who came the last week missed out on a lot. Maybe they should have come week 1. It seemed like there was a lot of energy around then.
- I enjoyed participating, the group interacted quite well and I look forward to putting into motion the concepts and ideas discussed and presented.
- I have gained:
  1) Ability to use manipulatives.
  2) Knowledge of software available.
  3) A support system.
  4) Ideas, ideas, ideas.
  5) A renewed enthusiasm for teaching math.

TEACHERS AS TRAINERS:
A STAFF DEVELOPMENT PLAN FOR SUCCESSFUL COMPUTER USE

Vicki E. Baer
Director, Learning Resource Center
College of Education, Box 3CUR
New Mexico State University
Las Cruces, NM 88003-0001

ABSTRACT

Providing schools with the hardware for emerging educational technologies is essential for school systems wishing to keep pace with contemporary developments. However, hardware cannot achieve its desired effects on student performance unless teachers are prepared to integrate new technologies into their already bulging curricula. This paper describes one school's preparation of its teachers for use of an instructional laboratory of networked computers.

The lab's success largely depended on educating faculty members to develop computer awareness, to select and evaluate appropriate software applications, and to use computers as tools to support instruction. The school's staff development plan focused on these three concerns.

Ten faculty members became a core group of teacher trainers responsible for instructing and encouraging their peers in the effective use of computers for content area instruction. By the end of the lab project's second pilot year, a majority of teachers were frequent users of the computer lab facility. Deliberate involvement of faculty members as models had provided an atmosphere in which teachers were not as hesitant to ask questions, were more willing to assume the role of "learner", and felt freer to explore the applications of this technology to their disciplines.

TEACHERS AS TRAINERS

When the Maryland State Department of Education announced that Boonsboro Middle School would be one of five sites to receive a networked laboratory of thirty-two computers for instructional use, a small team of faculty and administrators had already begun to plan for it. Their primary concerns for the lab's success were educating faculty members 1) to develop computer awareness, 2) to select and evaluate appropriate software applications, and 3) to use computers as tools to support instruction. The school's staff development plan focused on these three concerns. The underlying assumption? The lab's success depended on providing faculty members with the time and support necessary to use new technology to enhance the curriculum.

Site Administrator

The school district's superintendent created a site administrator's position for the computer laboratory to help ensure the success of the pilot project and the integration of computer use into the school's curriculum. The school's administrators selected the site administrator (the author). They chose a member of the school's faculty who had already established a professional rapport with her colleagues, who had administrative experience (department chairperson, program coordinator), and who had demonstrated a professional interest in educational computing (work toward a post-graduate degree in instructional technology).

Her full-time responsibilities included hardware setup, software installation and maintenance, keyboarding and word processing instruction, and teacher inservice instruction. She would act as a resource person for faculty and students as they used the lab for content area activities.

Teacher Core Group

The only networkable software available for the lab during the first year were word processing and keyboarding programs. As a result of this limitation, the school's first-year project would focus on using word processing to encourage writing activity in content areas. During subsequent years, specific content area software would be added.

Staff development began with a core group of ten teachers representing all curricular areas in the school. These teachers met in a four-day workshop prior to the lab's installation. First, they established short- and long-term goals for computer use. Short-term goals included:

- teaching all students fundamental keyboarding and word processing skills, and
- applying keyboarding and word processing skills to content-area writing assignments.

Long-term goals included:

- developing a program for writing in content areas usingcomputers as tools to facilitate the process, and

- establishing priorities for software selection and purchase:
  - basic skills development for remedial students,
  - thinking skills development for gifted students, and
  - general curriculum enhancement for all content areas.

Next, the core group learned how the new computer system operated. Not only did they adjust to a different brand of computer system (IBM PCjrs vs. familiar Apple IIes), they began to learn of the advantages a network system could offer them and their students: 1) all software and data files would be stored in a central file server, 2) a menu system would allow access to whatever programs the students or teachers chose, and 3) one high-speed printer would handle all document hard copies.

Then, the group learned how to develop word processing activities for their content area classes. They became familiar with a rationale for interdsciplinary writing, participated in some writing activities themselves, brainstormed writing activities for their disciplines, and used a model to design a sample activity for a class.

Finally, the core group examined software soon to be available for the network. There was a comprehensive basic skills package from Computer Systems Research which was matched to functional test competencies. There was a basic math skills package by Science Research Associates with a built-in management system. There was a complete biology and earth science program series by IBM for students reading at grade level or above. And there were several thinking skills programs by Sunburst which were instructive as well as fun. Teachers used the software, then completed evaluation forms for a later selection process.

Most important, this core group became teacher trainers. With the encouragement of the lab's site administrator, they became models for other faculty members. The group used the computer lab, encouraged other faculty to use the lab, and participated in staff development activities.

Staff Development Activities

Staff development activities were planned under one of several conditions: 1) new software became available for preview and evaluation, 2) faculty groups needed time to plan curriculum using newly installed software, or 3) faculty were introduced to methods for integrating computer use into their content area curricula.

A majority of staff development activities took place during the school day. Activities involving a portion of the faculty were scheduled with paid substitutes, with teachers on planning time, or with administrators covering for the participating teachers. When the entire faculty would be involved, activities were scheduled on a rotating basis or during a scheduled inservice day. (See Figure 1 for a sample inservice schedule.)

Each workshop provided several short breaks so teachers would not be forced to stare bug-eyed at their computer screens for hours on end. (After all, we wouldn't ask that of students, would we?) Having snacks available during breaks provided extra motivation and opportunities for informal sharing. Under the site administrator's supervision, core group members helped plan and lead activities and were present to help less experienced faculty learn the basic operations of new software and equipment.

Lab Scheduling

For the first ten weeks of each school year, all students (approximately 560 in grades 5-8) participated in a program of basic keyboarding and word processing skills. The site administrator scheduled and taught these classes, and all students participated in two thirty-minute sessions each week for ten weeks.

Teachers participated in the training with their students, either by attending each lesson as students or by helping students as the site administrator taught each skill. Meanwhile, the teacher core group had become the school's computer committee, and their first task was to develop a lab schedule for all teachers.

For the remainder of the first year, the lab schedule was opened after the training period on a first-come, first-served basis. Within two weeks, all time slots for the remainder of the year were filled. Teachers were willing to plan far in advance to bring their students to the computer lab for content area writing activities and to try out some of the new software packages which had arrived. This advance planning created one problem, however. Some teachers found no time available for their classes with the lab scheduled so far in advance. The computer committee set to work on a more equitable scheduling process.

During the lab's second year, computer committee members collected lab time requests from their department members for each grading period (every 9 weeks). At a computer committee meeting, the members shared their departments' requests, discussed scheduling conflicts, and finally filled in a scheduling form. Time conflicts were settled by consensus and creative organization. For example, some teachers agreed to combine their small classes to work in the lab at the same time. Others agreed to split class periods to complete short assignments. In their efforts to use the lab as much as possible, teachers worked together to use available time more efficiently. (See Figure 2 for a sample lab schedule.)

By the end of the lab project's second year, over half of the faculty (17 teachers) were regular computer users with some of their classes, and over three-fourths of the faculty (25 teachers) were occasional computer users. In faculty surveys conducted after each staff development activity, teachers consistently mentioned the value of the time they spent away from their teaching schedules to plan for computer activities. The following comments came from such surveys:

> Science teacher: "My time is more valuable than an hourly stipend to spend Saturdays and vacation days in inservice. Thanks for having this during school time."

English teacher: "We don't often have the opportunity to interact professionally with our colleagues outside the department. This workshop has given me a chance to get to know other teachers and their teaching needs better."

Math teacher: "Not only was the food good at this workshop, I was able to plan how I would use SRA Math with my classes. I can't think of when I would find that time otherwise."

Music teacher: "I never thought we [music teachers] would be using the computers to do composition activities. This inservice has convinced me to schedule time for music reports in the lab."

Social studies teacher: "COVERMAKER will give my students a welcome change of pace on 'Current Events Day'. I'm looking forward to their reactions."

Staff development activities had largely contributed to the teachers' willingness to use computers regularly to support their content area assignments. Deliberate involvement of faculty members as models provided an atmosphere in which teachers were not as hesitant to ask questions, to assume the role of "learner", or to explore the applications of technology to their disciplines.

FIGURE 1.
Basic Skills with CSR
Schedule of Participants and Activities

Periods 1-3: English, Social Studies, Special Education, Related Arts II
Coverage: Period 1--English: substitutes
Social Studies: substitutes
Special Education: coverage, RA I
RA II: coverage, RA I
Period 2--English: substitutes
Social Studies: none (planning)
Special Education: coverage, administration
RA II: substitutes
Period 3: English: none (planning)
Social Studies: substitutes
Special Education: coverage, administration
RA II: substitutes

Periods 5-7: Math, Science, Related Arts I
Coverage: Period 5--Math: none (planning)
Science: substitutes
RA I: substitutes
Period 6--Math: substitutes
Science: none (planning)
RA I: substitutes
Period 7--Math: substitutes
Science: substitutes
RA I: none (planning)

Activities (both groups):
1. Introduction to CSR Basic Skills package
2. Practice with content area modules
3. Planning for use with Functional classes

FIGURE 2.
BMS/METN COMPUTER LAB SCHEDULE, MAY

DAY DATE		MON (5-3)	TUE (5-4)	WED (5-5)	THU (5-6)	FRI (5-7)
PERIOD	1	Sci-17 Sci-12	Sci-20 Sci-21	Eng-24 SpEd-6	Math-31	Eng-27
	2	Eng-22 SpEd-4	Math-12 Math-17	Sci-29	Soc-29	Eng-22 SpEd-4
	3	Math-28	Eng-31	Eng-24 SpEd-6	Math-31	Soc-31
	4	Soc-29	Math-20 SpEd-7	Math-20 SpEd-7	Mus-30	Mus-30
	5	Eng-23 Math-8	Soc-27 SpEd-4	Soc-27 SpEd-4	Mus-28	Mus-28
	6	Soc-22 News-6	Eng-20 News-6	Sci-27 News-6	Sci-27 News-3	Eng-20 News-3
	7	Math-25 News-6	Math-25 News-6	Math-25 News-6	Eng-28 News-3	Soc-24 News-3

COMPUTING SKILLS WORKSHOP:
COMPUTER LITERACY IN THE CORE CURRICULUM AT CARNEGIE MELLON UNIVERSITY

Carol Scheftic
Center for Design of Educational Computing
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7644
scheftic@andrew.cmu.edu

Tracy Futhey
College of Humanities and Social Sciences
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-2830
tf01@andrew.cmu.edu

Abstract

A "computer-intensive environment" is one way in which Carnegie Mellon is often described. As many schools cope with developing various aspects of "computer literacy" among their population (students, faculty, staff, and administrators), Carnegie Mellon has mandated that all entering undergraduates participate in the Computing Skills Workshop (CSW) to prepare them for the array of computer activities that will be integral to their educational experience. This paper discusses which topics are included in CSW and why certain others have been excluded; it examines which lesson formats seem to work best for introductory computing activities and which were failures; it reviews the current status of the course and the plans for its future.

Introduction

As computer technology influences an increasing number of daily activities, Carnegie Mellon University (CMU) has made a deliberate choice to model, and even anticipate, these changes across all aspects of life within the campus community.

There are a number of reasons why CMU is an ideal setting for this endeavor. With an undergraduate population of about 4,000 students, it is large enough to need a systematic way of ensuring a common level of computing experience throughout the student body, yet still small enough to permit flexibility in trying out solutions to the problem.

There are well over 100 *kinds* of computers on the CMU campus, ranging from single-user personal computers to large, multi-user mainframe computers. For general campus use, the university is in the process of changing over from the use of a series of DEC TOPS-20 mainframes to a distributed computing system known as "Andrew" (Crecine, 1986; Morris, *et al.*, 1986). There are over 7,000 network connections on campus, through which personal computers or advanced-function workstations can access this distributed computing network. (In contrast, there are only about half that number of telephones on campus!) The university has equipped a number of rooms ("clusters") with computers, printers, and various specialized equipment (e.g., a digitizer) for use by members of the campus community, and personal machines can be connected to the network from dormitory rooms or offices. Dial-up access is available from off-campus housing.

Despite being relatively "rich" in hardware resources, the machines at CMU seldom sit idle. Personal computers located in the public clusters (Macintoshes and IBM PCs) and in private rooms (more Macs and PCs, plus various "clones" and some other types all together) get regular use for word processing, database and spreadsheet work, and programming activities. In addition, each student is entitled to a personal account on the main campus-wide system (now, that is Andrew), and the faculty members in certain courses may secure their students accounts on departmental computers or other specialized machines (e.g., at the Pittsburgh Superconputing Center).

There is extensive use of electronic mail (email) and bulletin boards (bboards) for all aspects of communication on campus. Assignments may be posted on a class bboard—there is no longer an excuse that a student didn't know what the assignment was. Questions can be sent to the teacher, and an answer returned, via email—there is no longer an excuse that a student couldn't get to the professor during scheduled office hours. Email and bboards can be used for social, as well as educational, announcements and discussions—and for research projects, such as consideration of changes in the nature of communication that result from their use. Electronic networks provide links both among the various systems on the local campus and with other such communities nationally and internationally—and students are expected to make use of those resources, as necessary, to locate information relevant to their studies.

For assignments, students are expected to present well formatted, clearly printed documents—as a minimum. They may be expected to use various other software packages in the preparation of their papers. Literature searches can be conducted on-line, with citations that can be "cut and pasted" directly into the students' papers. Some source material is also available on-line, so actual quotations can be inserted by the same process. Drafts can be sent, via email, to the teacher or to other students for comments and criticism, with a "marked up" version returned via the same route. Alternatively, the same type of interactions take place using files stored in certain central locations on the campus-wide network. Spreadsheets or statistical packages or other data processing programs are similarly available. In general, a high degree of "computer literacy" is assumed of students in classes after their first semester.

Initially, individual faculty members were responsible for insuring that all their students had the minimum computer skills required for the various activities related to their course. This might involve preparing a handout or organizing a workshop in an attempt to instruct the students in the necessary skills. This process, however, was ineffective and inefficient from several perspectives. Handouts were generally inadequate for conveying the complexity of the interactive processes involved to any but the few students who already had some computing experience. There was seldom any "extra" time in a course that could be used for lessons on the various computing skills required, yet scheduling "extra" sessions to cover them caused other difficulties: determining an appropriate time that did not conflict with the variety of student schedules, locating and scheduling an appropriate facility, expecting an individual faculty member in a particular discipline to provide instruction on computing skills that were often at or near his or her own limits of experience, and so on.

The ubiquity of computing resources thus presented a clear new problem: how to assure a basic, common level of experience among all students? One solution could be to develop a single introductory course, one that would be driven by campus usage and needs, not by its own internal desires, that would provide this shared basic experience. This strategy, however, raised new questions. Was there really a common need among students as diverse as music majors and civil engineers, psychologists and physicists? Was it better to provide extensive drill and practice on a limited set of specific commands in particular software applications, or to provide a broader view of computing resources and techniques for independently determining the necessary information?

1. "Computer Utilization"

Of course, it is only hindsight that makes it possible to state the questions so directly. Having set the stage, as it were, let us step back a few years and describe the process in a more evolutionary way.

Early in the 1980s, the situation looked somewhat different. There was already a good variety of hardware and software available on campus, and there was beginning to be a "critical mass" of people—faculty, staff, administrators, undergraduate and graduate students—who were actively involved with computers for various aspects of their regular work. There was an introductory programming class (Chandhok, et al., 1985; Scheftic & Goldenson, 1986) that was required of students in most undergraduate majors, although (1) it could be taken at any time prior to graduation and (2) it covered only a specific set of problem solving and programming skills, but few of the more general aspects of "computer literacy."

There was concern that the historical division between the "arts" and the "sciences" would become even more pronounced as various technological changes increased. Already, some of the students in the College of Humanities and Social Sciences (H&SS) were complaining of being at a disadvantage compared to the science and engineering students. In many cases there existed a course available to, and possibly even required of, students from all majors, yet only some of the incoming students would have the prerequisite skills—not in the academic discipline under study, but in the ancillary area of computing.

A. First Year - AY 83-84

In response to this need, a course known as "Computer Utilization" was created. It was offered to all incoming freshmen in H&SS during the 1983-84 academic year. Approximately 300 students took the six-week course during the fall semester in 1983, in either the first or second half of the semester.

The format of Computer Utilization was based on a weekly, two hour lecture. During the lecture, students were given regular "assignments" that they were to complete outside of class, in public computing clusters in several locations across campus (most of them open 24 hours each day, 7 days each week). The assignments were "suggested." That is, they were collected and counted, but not actually graded. Undergraduate students reviewed the assignments according to directions provided by the course lecturer. The reviewers were available to the students as a source of information outside of the lecture, though this was through informal, student-initiated contact. The course was offered on a pass/fail basis, and a student's final status for the course was based entirely on their performance on a proficiency exam.

In the first year, the course activities addressed only the Digital Equipment Corporation TOPS-20 systems that then provided the bulk of the common campus-wide services. As outlined in Table 1, the topics included communications, document production, a statistical application package, and various operating system features.

B. Second Year - AY 84-85

A large experiment, to be sure, but that first year was determined to be a reasonable success. So it was repeated the following year, in the Fall Semester of 1984. Several changes were made, in an attempt to improve the course. First, it was extended from six to ten weeks, and personal computing was discussed in addition to mainframe computing. A summary of the specific programs included in the second year's course is given in Table 1.

Second, undergraduate TAs were scheduled to assist students in the public clusters. That is, in addition to the regular "Cluster Monitor" who supervised each such facility, there was also a group of students who held regular "office hours" in the clusters, assisting students with any problems directly related to the Computer Utilization assignments. Lastly, the assignments were now required; they were collected and graded each week. The proficiency exam remained, and it was similar in style and content to that given in the first year. The course was still pass/fail, but now the assignments and exams each contributed 50% toward determining a student's status. The format of the first two years of Computer Utilization is also summarized and compared in Table 1.

C. Looking Forward After the First Two Years

As with the first year, the second year was again judged a reasonable success. Indeed, there were beginning to be requests from faculty in other colleges, not just those in H&SS, to have the course available, in whole or in part, as a requirement for their students.

As a result it was decided that for the following year, the course would be required campus-wide, of all students, from all colleges and majors. A review of campus needs was conducted, in order to determine the most appropriate mix of topics to include, and to improve the sequence in which skills were introduced. Individual sections would be established, with assigned Teaching Assistants. In this way, students and TAs would get to know each other, increasing the accountability over that possible with the seemingly random encounters in the public clusters. With scheduled times in laboratory sections, it would be possible to assure that the work turned in by an individual was actually done by that person. Similarly, with TAs assigned to a section, they could better be reviewed, rewarded for good work, and held accountable for any misinformation they conveyed. Finally, the course would be based within the core curriculum and would be designed to meet the objectives of that program. For example, each section would contain a heterogeneous mix of students, instead of assigning blocks of students from the same department together.

2. "Computing Skills Workshop"

The result was the course known as Computing Skills Workshop (CSW). Although details of its evolution will be described in subsequent sections, it has always had the following goals and associated processes:

- To introduce students to the hardware and software that they are most likely to use during their time at CMU, use the same hardware and software as available in the public clusters.
- To have students familiar with these resources as early as possible in their time at CMU, require the course of all freshmen during their first semester on campus (while keeping the time investment to the minimum possible amount so pressures from other classes do not interfere).
- To give students a basis for comparison among hardware and software resources (e.g., which application to use for an assignment, what computer they may want to buy, etc.), introduce systems and packages that can do the same thing; compare and contrast features and ease-of-use.
- To help students to become familiar, to feel comfortable, with various aspects of computing, emphasize techniques for *learning how to learn* about various hardware systems and application packages, rather than covering any individual one in depth.

A. First Year - AY 85-86

From its beginning, CSW has been a required course for all CMU freshmen. In its first year, just over 1,000 students were assigned, in groups of at most 40, to a total of 30 sections. There were two "sessions" lasting 7 weeks, so that the sections could be divided about equally between the first and second half of the semester. Table 2 provides a summary of the topics and format of CSW, outlined to permit comparison of the designs in different years.

A few additional sections of the course have also been offered each year during the spring semester, where the students consist of a mixture of (1) those who failed it in the fall, (2) older students who started before the course was a requirement but who wanted the experience, (3) students who transferred to CMU from other schools and had therefore not been scheduled in as freshmen, and (4) staff members, particularly recently hired ones.

Students spent 3 hours each week in class. Half of that time was spent in a large lecture hall, equipped with a computer whose image was projected onto a large screen at the front of the room. The lectures were given by the course coordinator. Each lecture was given twice per week, each time to approximately half of the students enrolled during that particular session.

Students were also assigned to a "section" during which they spent a further 1.5 hours each week in a hands-on computer laboratory setting. These sections were scheduled from 10:30 am through 10:30 pm, Monday through Thursday, and there were additional hours scheduled on Friday afternoons for open help sessions. The CSW course "took over" one of the previously "public" clusters for the entire fall semester. The room was located in the basement of the main library building, and contained 40 Macintoshes. A device known as a MacCharlie was added to each machine, thus enabling it to simulate an IBM PC and to run actual IBM PC software. Each machine was also connected to one of the campus networks, thus enabling it to be used as a "terminal" to the mainframe (TOPS-20) computers covered in the class.

For the 40 students assigned to each section, there were four undergraduate Teaching Assistants (TAs). Students were expected to attend the lecture, read information contained in weekly newsletters, and come prepared to complete their assignments in the 1.5 hours allotted for their lab section. Assignments were graded by the TAs. The course itself was pass/fail, and a student could pass the course by completing all of the assignments. There was an exam, but it was optional: it was intended primarily as a mechanism by which students could "make up" an assignment that they had missed.

One final item concerning the impact of the course: it even forced a change in the course numbering scheme used by the Registrar's Office. With a relatively small student body, only one alphabetic character had been allowed for the section designator with a given class number: no one had imagined that there would ever be more than 26 sections of a single course. Clearly it had to be changed, with CSW having 30 sections in that first year and even more in years to come!

Evaluation of CSW's First Year

In retrospect, it was generally agreed that "'Twas better to have offered the course, and learned, than never to have offered it at all..." Buoyed up by the relatively early success of Computer Utilization, it was somewhat of a surprise to discover that the course could not simply be expanded indefinitely.

For example, morale became a real problem. As soon as students discovered that grading was done in laboratory sessions, the attendance at lectures dropped. But the majority of the information needed to complete the assignments was being presented in the lectures; by not attending the lectures, many students came unprepared to labs. As a result, TAs had to teach what had been covered in lecture. Students who had gone to the lectures were delayed, while students who missed lectures received poorer instruction. Lab time was effectively reduced, while the TAs were feeling overwhelmed at actually having to teach concepts, rather than assisting with minor difficulties. And in addition to a variety of technical problems with the MacCharlie configurations, a number of students had real difficulty with the level of abstraction required to understand the use of a single machine three ways: as a Macintosh, to simulate an IBM PC, and as a terminal to a mainframe system. The TAs became frustrated while, at the same time, many students stopped taking the class seriously.

On the other hand, the course was certainly far from being a disaster. Even with all the problems, 95% of the students passed it. Of those who failed, it appeared as if apathy was far more important a reason than was actual difficulty with the lessons. While it was decided to continue to offer the course in subsequent years, it was also obvious that restructuring was necessary.

B. Second Year - AY 86-87

In its second year, CSW was again offered in the fall semester, to approximately 1,100 new undergraduate students. The course was expanded to a full 13 weeks. There were now 70 (yes, seventy) sections, with 20 students each. Each group met for 1.5 hours in a computer laboratory facility. The large lecture sessions were eliminated: all teaching and practice were done in the laboratory sessions.

Although the course was expanded to 13 weeks, instead of 7, the class time per week was reduced from 3 hours to 1.5. A new fall schedule was established to accommodate 70 sections of 1.5 hours in length. Classes were held in two facilities at a time, during the following hours:

Monday through Thursday	08:30 a.m. - 09:00 p.m.
Friday	08:30 a.m. - 04:30 p.m.
Saturday	10:00 a.m. - 01:30 p.m.

There were 20 students in each section, with two undergraduate TAs. (Additionally, there were 9 sections offered in the Spring Semester 1987, having a combined enrollment of slightly over 100 students.) To maintain consistency among the 70 different sections, the basic lesson was presented in a 20 to 30 minute slide-tape show. The remaining hour or so was designated as a hands-on practice session. There was an assignment each week, designed so that the majority of students could complete it during their section meeting. Assignments were collected and graded by the TAs, and returned the following week. Letter grades were assigned in the course for the first time: it was no longer pass/fail. It was worth 3 units (equivalent to one "credit" at many colleges).

As part of the reorganization, the curriculum was completely reviewed. In addition to the mainframe (on the TOPS-20s) and personal computing topics (IBM PC and Macintosh), a unit on distributed computing was added. This new unit included using advanced-function workstations, such as the IBM PC RT, the Sun 3, or the VAXstation II, connected to the campus-wide Andrew network (see Table 2).

Balancing topics and facilities, it was decided to divide the course into four equal-length "quadrants" and the students into two "groups." Both groups would have the same first week, describing the course structure and requirements. Beginning with the second week, the groups would be differentiated, although each group would first cover a system that provided communications capabilities, including electronic mail and bulletin boards. After all the topics for that system had been covered, there would be an exam. Then the group would move to one of the personal computing systems, after which there would also be an exam. Halfway through the semester, the groups would trade places and cover the remaining two systems. This arrangement is outlined in Table 2.

In designing the course content, attention again had to be paid to the available facilities. In the early stages of this revision, plans were drawn up for a new facility designed with the particular needs of CSW in mind. A problem arose, however, in finding appropriate new accommodations for the people occupying the designated space, and construction on the facility was postponed indefinitely. Instead, it was necessary to use three separate facilities. The mainframe and IBM PC quadrants were taught in the University Computing Center building at the north end of the campus. The distributed computing quadrant was taught in a workstation cluster in a remote back hallway of one of the engineering buildings at the west end of the campus. And the Macintosh quadrant was taught in a room in the basement of the main library, in the south-central area of campus.

Some simple mathematics are useful in describing the succession of events. There were 13 full weeks in the semester's schedule. One week was taken up with the "common" introduction and orientation, leaving 12 weeks for covering the material in the 4 quadrants. That meant that there were 3 weeks available for each quadrant, with two for the "lesson" and one for the exam. The logistics of swapping facilities for each quadrant meant that no other design was really possible, no matter how "uneven" that left the pacing.

With two weeks of lesson for each quadrant, there were eight, weekly, graded assignments in all, and four, quarterly, graded exams. All were designed so that a student should be able to complete them during the time allotted for CSW. For the assignments, however, a slower student could complete them later using public cluster equipment.

To supplement the slide-tape shows, a "book" was designed to accompany the course (Computing Skills Workshop: 1986-87). It was a thorough, detailed manual on using the various computing tools introduced during the course. It was intended to be an instructional and reference manual that would serve student needs both during Computing Skills Workshop and throughout their tenure at Carnegie Mellon. On a weekly basis, supplementary handouts were provided that pointed out various items of particular interest. For example, "jargon sheets" evolved as one particularly useful supplement. They were designed to provide the students with a list of key terms covered in the slide presentation and to serve as a reference if part of slide show was missed or misunderstood.

The exams had to be completed in class. They were designed to last one hour. Students were allowed to refer to neither the course book nor their notes. The exams were laid out with two parts and, for each part, the students had to use a different color pencil (provided) and answer in a different area of the exam paper. During the first 30 minutes, the exam was "closed computer." For a student to receive any credit for a question, there had to be *something* in the space for the answer, even if it was simply "I don't know." For the second 30 minutes, the students could work on-line, try out the answer they had indicated in the first half, use the system help facilities to get further information, try out other alternatives, and indicate a final answer. Their response was marked correct if any part of the answer was correct. That is, there was no penalty for the first answer being wrong, there just had to be an indication that the student had considered the question before using the computer to "try out" the answer. The design was intended to help students see how much they actually knew outright, and how much more information they could discover on their own. It was intended to encourage them to become familiar with the on-line help features of different packages, and to try experimenting with different options when they were unsure. In other words, it was intended to be as much an instructional activity on its own as it was an evaluation mechanism.

Consistent with that design, a return was made to having the assignment of letter grades weighted towards the exams. Furthermore, that helped significantly to reduce the potential for cheating. Since the assignments were actually relatively simple for an experienced computer-user, weighting the grade toward the in-class exams reduced the possible contribution that could be made by a friend completing a student's assignment outside of the class time. Students who entered the course with little or no computing experience, but who had done their own assignments, were likely to score well on the exams. Students who already had some familiarity with one or more of the systems, however, could do reasonably well without having to spend a lot of time on what they might consider make-work assignments.

Evaluation of CSW's Second Year

Overall, the second year of CSW was much better than the first, despite the additional hurdle of having the person who had administered Computer Utilization and, later, CSW resign in order to accept a position at another university. On the other hand, it was still far from a simple undertaking.

Administering three separate facilities was, quite simply, unmanageable. The necessity of a separate, dedicated facility for a project of this magnitude was undeniable.

Although the slide shows did go a long way toward guaranteeing consistency among the sections, they clearly needed more design work and further production. Seen in one order, the progression of topics was very reasonable; but the "quadrants" were seen in different orders by the different groups. Although the lessons reinforced each other, providing comparisons and contrasts among the different systems and software packages, it could not be assumed that an individual viewer would see a particular show before or after most others. Sequencing was therefore a tremendous concern. Another set of significant issues concerned questions of depth vs. breadth and theoretical vs. practical considerations.

These issues were relevant, not only to the slide-tape shows, but also to the book that had been written to accompany the course. Technically, the book was quite sound, although not all of the students were able to appreciate that. While some potential problems seemed obvious, it was also the case that none of the reasonable alternatives appeared any less danger-prone. For example, eliminating redundant information could help to make a slide-tape segment less boring and the book smaller and less intimidating, but it also introduced sequencing dependencies. Similarly, a theoretical discussion of standard file manipulation operations (read, save, delete, copy, rename) could give the students a framework for approaching the learning of a new system, but the need for that framework could be missed by the first-semester freshmen who were motivated by discovering how much they could churn out in a single lab session and not by the fact that these were simply "standard" operations.

Finally, there were concerns about the exams. There were seventy (70) different sections taking the exams over the course of a week. Particularly with the close quarters in freshman dormitories, it was unlikely that students taking the exam at the end of the week would not have heard discussions among students taking it earlier, even before consideration is given to the possibility of outright cheating by passing information along. Giving the same exam to 70 different sections was out of the question, but developing and coordinating 70 completely different exams was also a nightmare. Clearly, a reasonable compromise was needed.

C. Third Year - AY 87-88

In its third year, CSW was again required of all freshmen during their first semester. By the fall semester of that year the size of the freshmen class had grown to almost 1,200 students. Students were divided among 70 sections, with 20 or fewer students in each section.

One big hurdle was overcome with the opening of a facility designed specifically for the training of students in the Computing Skills Workshop and available exclusively for CSW activities during the fall semester. This facility, established in a remodeled wing of one of the oldest buildings on campus, consists of a 50-seat lecture room and two 20-seat computing laboratories equipped with computers appropriate for either Distributed Computing or Personal Computing. The Distributed Computing lab contains 20 Andrew machines (specifically, Sun 3 workstations) while the Personal Computing lab contains 20 *each* of Apple Macintosh SEs and IBM PCs (i.e., both types of personal computer sit side-by-side at each station). The existence of a single, dedicated facility[1] greatly reduced many of the problems that arose from "borrowing" public clusters. It also served to focus administrative efforts on a single location.

Sections met weekly in 1.5 hour sessions and there continued to be 13 of these meetings. After one common introductory session, students studied one of two "types" of computing:

- Distributed Computing,
  on the Andrew environment, or
- Personal Computing,
  on the Apple Macintosh and the IBM PC.

The Distributed and Personal Computing portions were each 6 weeks in length. Following the first week's common introduction, half of the students began studying Distributed Computing while the other half began studying Personal Computing. After 6 weeks the groups were switched to the other set of topics. Topics related to mainframe computing were eliminated from the curriculum, as the TOPS-20 systems were being removed from service at the university.

The topics introduced in the Personal Computing section were largely the same as had been introduced in the previous year, with the addition of several applications that allowed access and use of Andrew's electronic communication and network services from the IBM PCs. The topics introduced in the Distributed Computing section included electronic communication, two forms of document production, and an introduction to the university's library information system. Also included was a brief introduction, without assignment or examination, to the cT programming environment (CMU Tutor, 1987; Sherwood & Sherwood, 1988).

A typical weekly session would begin in the lecture facility with a slide-tape show. This form of presentation continued to be the primary means of introducing the weekly lessons, although a completely new set of shows had been produced. Detailed scripts were produced, with their primary author being one of the undergraduate teaching assistants who had been involved with the course for several years. Slides were taken and sent out for developing. Synchronized cassette tapes were produced in-house after the slides were returned. Two videotapes were also used during the semester. One was produced by the University's Instructional Technology department covering the on-line, interactive Library Information System. The other (CMU Tutor, 1987) was produced jointly by the University's Center for Design of Educational Computing and International Business Machines (IBM) covering the CMU Tutor Programming Environment (now known as cT).

After the half hour media show introducing the weekly topic, there was a one hour hands-on lab in which students applied the skills that had been described and demonstrated in the lecture room. Teaching assistants provided additional help as needed. They were encouraged to help the students find out for themselves the answers to their questions, and discouraged from giving them the specific answer immediately with no explanation or attempt to stimulate further consideration of the question by the student. While two teaching assistants for a class of about 20 students was generally adequate, there were still occasions when the two TAs could not effectively handle all the questions generated in a particular section.

---

[1] Calling it a "dedicated" facility is only slightly misleading. It is really an additional public cluster, with the exception that it is booked 14 hours/day Monday through Friday and 3 hours on Saturday during the fall semester for CSW classes. It is, like all clusters, open 24 hours every day, and is publicly available whenever not in use by a class. This gives the campus one more public facility; it also means that the hardware gets that much more use and needs that much more maintenance.

In addition to the introduction provided by the media presentations, students were supplied with a text for the course. A new text was required since there was a significant difference in the topics and applications compared to those introduced in the previous year. Since changes were being made, the design was also changed to make new text more modular, less like a manual. It focused on the specific information that students required in order to receive a basic introduction to common aspects of the system and to the applications introduced in CSW. Supplemental course materials included jargon sheets, containing basic terms introduced in the slide-tape shows, and supplemental handouts, giving further detail on certain aspects of a system or application. These materials were all provided as support for completing the assignments and exams.

In each half of the course there were:

- weekly assignments, set each week and due the next (a total of 5 assignments per half), and
- an exam, administered on the sixth week of the half.

The assignments were designed so they could be completed by the students during the lab time for their section. An additional week was allowed before they were due, however, for those situations in which students experienced difficulty in finishing an assignment during lab. The exam was administered during class time and students were required to complete it within an hour. Initially, the format of the exams was identical to the format as described in the second year. During the spring semester of the third year, however, the exams were redesigned. To include more emphasis on using the systems and applications and less on specific command lines or keystroke manipulations, exams no longer consisted solely of questions that asked the student to describe a specific keystroke manipulation or system command. They also included exercises that tested the skills the student used in completing assignments.

For the first time exams were generated by a program, XamMaker, rather than by hand. The XamMaker program was begun by one of the undergraduates while working as a CSW TA. The student then completed it as a project in the CDEC Scholarship Program in Educational Computing; the slide-tape show production and the book revisions were also CDEC Scholars' projects (Scheftic, 1988).

Evaluation of CSW's Third Year

In general, the third year has proven to be more successful than previous years. The new facility that the course occupies is a major factor in this success. Also, the new exam format was well received by students and is clearly a more accurate way of testing the skills that they have acquired. The distribution of grades continued to be high (see Table 2) with nearly all of the students who regularly attended class receiving a grade of A or B.

Still, a number of challenges do remain, most notably:

- further attention to the production of the media-based presentations,
- possible development of optional advanced material for more experienced students, and
- more rigorous training of the undergraduate teaching assistants.

The slide-tape presentations effectively accomplished their purpose—a basic introduction to the weekly topic—but they did not clearly convey all of the information that a student might need in order to use the specific application that was introduced. In retrospect, too much elapsed time was spent on design, with the result that insufficient time remained for the actual production of the shows. The production quality suffered and the effectiveness of the presentations was reduced. If students did not pay careful enough attention to the material being introduced, particularly if they missed a key point early on in the show, then they could never fully grasp the information that they did see. More professional productions become crucial in holding the student's attention.

In addition to improving the production quality of the presentations, there are other advantages to considering an alternative presentation media for some of the topics. Systems like the Andrew and the Macintosh are so visually oriented that a static slide presentation does not effectively convey many of the features nor even the basic use of such systems. A video presentation could provide a better introduction in a shorter period of time, thereby increasing the hands-on time for students. (A side benefit is that videocassette players are much easier for the TAs to operate than are synchronized multi-projector slide-tape shows.) Two "pilot" videos used in this year were very well received by the students.

3. The Future of CSW

Although the comments on the evaluation of each year, above, have pointed out specific items of concern with the particular design used that year, there are a number of general issues that appear on a recurring basis. Before describing the plans for the next academic year and beyond, these recurring themes will be discussed.

A. General Issues

A course of this magnitude requires creative thinking about its design. For example, the current 13-week schedule covers the longest possible span since there can only be as many sessions as the smallest number of any given "day" (e.g., Monday) in the semester.

Hardware reliability is a constant concern, even with the new facility. It receives, quite simply, extraordinarily heavy use. It is used by novices, students who may still be somewhat naive until it is too late concerning how to take reasonable precautions while using the equipment. It is used in a continuous stream, by students who are learning to take advantage of the widest possible variety of features. Its use is intended to encourage enthusiasm, so students are likely to come back at night or on the weekends and work the machines even harder. It is used in an environment where everyone relies heavily on the available computing resources, placing a high demand on the support groups responsible for maintenance and repair across campus. It is used in a situation where the tight schedule magnifies the effect of any delay in repair, since it prevents some students from participating in the activities as planned.

System reliability is also critical, especially on Andrew and the associated networks (or any "remote" system, such as the discontinued TOPS or the in use but not yet covered VAX systems). For example, if the system is down for just one hour, then one whole section will miss its only opportunity for completing on-line tasks. On "lesson" days, that is unfortunate, although the hands-on practice can still be completed later in a public cluster. For "exam" days, however, there is no reasonable alternative, since the fully packed schedule and facility swapping prevents delaying that section's exam at all. The same is true whether the down-time is caused by a system failure or by preventive maintenance: if a section has to be cancelled, there is neither time nor facilities left in the schedule to make it up.

On the Andrew network, particularly, since it is an "experimental" system still undergoing development, further risks involve the possibility of suddenly finding that a new version of system or application software has occurred overnight. Then the slide shows may not match the on-line interaction, the TAs will have never seen the software themselves and therefore cannot anticipate student difficulties, and many students end up confused and resentful. Even worse, if the change occurs during an exam week, students are forced to take an exam on a version of software that they have never seen before. Even though one of the course's goals is to enable students to cope with such new releases and even with totally new software, encountering such a change in the middle of an exam week causes much frustration among students, TAs, and course administrators in determining fair grades for sections whose exams were given before vs. after the change.

Considerable efforts have been made, on both the part of the course coordinator and the the system maintainers, to provide a stable environment on the Andrew system, timing software releases and updates so as to cause minimal disruption to CSW students. In spite of these efforts, there continue to be problems related to system reliability and availability. Loss of communication with the remote file servers results in system inaccessibility. Students have also been displeased with the slowness of the system and the bugs that they encountered. Robustness and thorough testing of the system remain very important, particularly in the CSW setting where access time is limited in the first place and students' inexperience makes it take longer to understand and resolve bugs.

Issuing user-ids to all the freshmen immediately has severely strained the capabilities of various academic computing groups. All of the sections start at the beginning of the year, so all 1200 freshmen need user-ids immediately. One might think that, with the rotating sections, only half of the students would need user-ids right away. But with the huge number of section changes generated during the first week of class through the add-drop process, it is unrealistic to try reliably predicting which system any given student will see first. With the complexity of the schedule, there is no way to guarantee any reasonable way of making up a lesson for students who are not yet authorized; and with the severe drain on so many resources at the start of a semester, it is difficult to guarantee when a user-id can be generated for a student once it is discovered to be missing.

Many of the previously described logistical problems relating to the magnitude of the course remain, but the effectiveness in handling them has improved with experience. Scheduling such a large number of sections and planning around holidays and breaks continue to pose problems as do the coordination of 35 teaching assistants and their assignment to the 70 sections. In theory, each section of 20 students is to be staffed by two teaching assistants. But finding students free to work as TAs at the more popular times is difficult: those sections most likely to be completely filled invariably occur at those times when TAs are least likely to be available.

Identifying and training TAs is another large and important task. Recruiting is done early in the spring semester. Applicants are selected based on superiority in interpersonal skills and competence in various computing activities. That is, computer wizardry alone is not enough. The TA need only be able perform the skills required of the students: to use a variety of tools and to know how to locate information when asked a reasonably challenging question, but not necessarily to know every answer immediately. However, the TA must be able to make the students feel comfortable with the new technology, to encourage all students to work to the best of their abilities, and to effectively convey the answers to questions. By identifying TA candidates early in the spring semester, they have time to become familiar with any new systems or packages. TAs do attend a one-week intensive training session the week before the fall semester starts. Organizing and supervising such a group of undergraduates is a challenge, to be sure. But they provide an exceptional opportunity as well since, being undergraduates, they can remember the fears associated with the new environment and they can provide believable anecdotes about the ways in which various skills will be useful to the students.

Freshmen who come to the university with considerable computing experience, or who acquire that experience very quickly in other settings, are sometimes frustrated with the introduction that they received in CSW. Since the course is targeted to the begining user, the more advanced students sometimes feel that they are wasting their time in a class that is trying to teach them things that they *think* they already know.

On the other hand, students who are less advanced and who lack a basic understanding of what is presented in the shows are unable to complete their assignments without extra help and attention from the teaching assistants. As a result, the teaching assistants are frequently called on to provide a student with an introduction to some application or topic. Although the TAs' preparation has included mastery of the skills introduced in the course and several sessions dealing with topics such as ethics and motivation, they are not trained teachers. Since some students require more of teaching assistants than expected, the TAs often spend too much time helping a few students while questions from other students go unanswered. Although one possible solution involves increasing the number of TAs per section from two to three, other alternatives include improving the media-based presentations, improving the TA training, and reducing section size by offering ways for the more advanced students to "place out" of some or all of the lessons.

Even if we were to find a way to resolve all of the above issues, one more would remain. Each year, the course is thoroughly reviewed, and changes are made as needed to the media-based presentations. In practice, however, with the constantly changing system and application software, it is basically necessary to rescript and reproduce the shows *every* year. While this is expensive, that is not the only concern. It is difficult to anticipate very far ahead the format and curriculum for a coming year. Many aspects of the course are determined by system features, and development on the Andrew system is not yet complete. Furthermore, the personal computing section of the course introduces packages that are publicly supported on campus and decisions about supported packages for next year may not be finalized until a few weeks before the semester begins. This uncertainty seriously limits the design and production possible for the media presentations and the course text.

B. The Future - AY 88-89

As of June 1988, what is known is that there will continue to be an introduction to Andrew, the Apple Macintosh, and the IBM PC. Although the specific applications may not yet have names, their functionality is known. Students will be introduced to several applications for producing documents, at least one graphic editor, at least one spreadsheet application, and several methods of electronic communications. There will certainly be additional topics introduced in CSW, but the backbone of the course will remain.

The use of media-based presentations will continue although the specific media may be video or some combination of video and slide-tape shows. One key in the presentation of lesson materials is to showing a setting that closely resembles what the student will experience. Increasing the accuracy of the visual presentations will reduce the amount of verbal description that is necessary, thus providing more time for hands-on experience and reducing the amount of extra assistance required by individuals. Since interactive video is being considered for use in later years, there is an additional advantage in moving to video production in the near-term as well. Although a variety of resource limitations will mean that this shift must occur somewhat gradually, the development of video scripts is ready to begin.

Another major development effort currently planned is a complete revision of the text. The new edition will contain high-level descriptions of computing, and the systems and applications introduced; it will also have specific reference information. Supplementary material will describe aspects of the university computing environment that are not addressed in the course. The goal is to provide the student with a computing reference useful beyond the scope of CSW.

Examinations will continue to test a combination of acquired operational skills and understanding of specific commands. In order to maintain accurate and unbiased testing conditions, the database of exam questions will be expanded to include more questions and a range of operational exercises.

C. The Future - AY 89 and Beyond

CSW was instituted as the university's first core course because of an increasing reliance on computers within the university and because of an overall lack of computing skills on the part of entering students. Each of the three years in which CSW has been required has seen an increase in the number of freshmen who come to the university with computing experience. This additional exposure to computing has resulted in diversifying the level of student experience so greatly that it becomes difficult to give all students a common introduction. Many students still have little or no experience with computers but an increasing number do have experience, ranging from a limited amount on one system to an in-depth understanding of several systems. The increasing depth and breadth of computing experience suggests tailoring several computing introductions to different groups, according to their experience and needs.

There are several alternatives for restructuring CSW so that it more fully meets the needs of all university freshmen. Students might be grouped on the basis of their computing experience and the lessons tailored accordingly. Another possibility would separate students by disciplines and offer specialized versions of the course to each of these "interest groups." Yet another alternative would be to increase the modularity of the course, giving students the option of completing the required assignments early and moving on to more advanced or special-interest topics.

On-line tutorials present another possibility. A tutorial could supplement, or even take the place of, the presentation. It is in this context that the prospects of interactive video are most interesting. By taking advantage of interactive video to introduce the specific systems and applications, lectures could be used for more general discussions of general concepts and of more advanced software and systems outside the hands-on realm of CSW.

Each of these alternatives offers certain advantages over the current structure of the course but each also has some disadvantages. When CSW was introduced as a core course, one of the desirable effects was the integration of students from all majors in a common course. Students who would otherwise have little or no occasion to interact with one another would attend the same class and would complete the same assignments. Tailoring different versions of the course to students with different skill levels or from different colleges might rebuild those same barriers that CSW has tried to remove.

Aside from any issue of what role CSW should play in integrating students, however, there are other factors that make the above alternatives difficult to implement. The computing environment at Carnegie Mellon (and elsewhere) is volatile. As a result, course materials are seldom completely accurate for longer than a semester or, if you're lucky, two semesters. This changing environment makes curriculum design and production a time-consuming and never-ending process. Developing one such curriculum is a very large undertaking; developing several could be overwhelming.

It does seem like a task worth trying, however, if we are to support our students' use of the advancing technology.

Acknowledgements:

First of all, it was John Stuckey who understood the need and started us all on the path to solution that has become known as Computing Skills Workshop. Preston Covey, wearing his several hats as Vice Provost for University Studies and Director of the Center for Design of Educational Computing, has provided both an intellectual home and the resources that feed CSW. John P. Crecine, while Senior Vice President of Academic Affairs, was particularly instrumental in providing the computing facilities for the course. Stephen Fienberg, Dean of the College of Humanities and Social Sciences, has generously permitted his staff members to lead the efforts. The Division of Academic Services has been particularly helpful with overseeing the facilities taken up by the course. Space does not permit naming all of the undergraduate Teaching Assistants who have contributed to CSW's success. Without their dedication this course would not be possible.

About the Authors:

Carol Scheftic is a Research Scientist with the Center for Design of Educational Computing at Carnegie Mellon. She has served as advisor and consultant to Computer Utilization and Computing Skills Workshop, in various formal and informal ways, since its beginning. Most recently, as Director of the CDEC Scholarship Program in Educational Computing, she has provided support (both financial and advisory) to many of the students who worked as instructional designers and media developers for CSW.

Tracy Futhey is Director of Computing for the College of Humanities and Social Sciences at Carnegie Mellon. She was a Teaching Assistant in the early stages of Computer Utilization, assisted the Coordinator for several years, and is currently the Coordinator of the Computing Skills Workshop. In this role, she is responsible for course design and development and for directing all the activities of the students teaching assistants.

Bibliography

Computing Skills Workshop: 1986-87. Pittsburgh, PA: Center for Design of Educational Computing, Carnegie Mellon University. (Developed by John Stuckey).

Computing Skills Workshop: 1987-88. Pittsburgh, PA: Center for Design of Educational Computing, Carnegie Mellon University. (Developed by Linda Guthrie Branagan and Tracy Futhey).

Computing Skills Workshop: 1988-89. Pittsburgh, PA: Center for Design of Educational Computing, Carnegie Mellon University. (Developed by the CSW Development Team).

CMU Tutor. (1987). Pittsburgh, PA: Center for Design of Educational Computing, Carnegie Mellon University. (This is a videotape, produced by CMU and IBM).

Chandhok, R., Garlan, D., Goldenson, D., Miller, P., & Tucker, M. (1985, July). Programming environments based on structure editing: The GNOME approach. In *Proceedings of the 1985 National Computer Conference*. Chicago, IL: IFIPS Press.

Crecine, J. P. (1986, February 28). The next generation of personal computers. *Science, 231*, 935-943.

Morris, J. H., Satyanarayanan, M., Conner, M. H., Howard, J. H., Rosenthal, D. S. H., & Smith, F. D. (1986). Andrew: A distributed personal computing environment. *Communications of the ACM, 29* (3), 184-201.

Scheftic, C. (1988, June). Students as developers of educational computing projects: Experiences from Carnegie Mellon University. In *Proceedings of the National Educational Computer Conference (NECC '88)*, pp. 312-321. Eugene, OR: International Council for Computers in Education, University of Oregon.

Scheftic, C., & Goldenson, D. R. (1986, June). Teaching programming method and problem solving: The role of program environments based on structure editors. In *Proceedings of the National Educational Computer Conference (NECC '86)*, pp. 231-236. Eugene, OR: International Council for Computers in Education, University of Oregon.

Sherwood, B. A., & Sherwood, J. N. (1988). *The cT Programming Language*. Champaign, IL: Stipes.

Stuckey, J. (1986). The Computing Skills Workshop at Carnegie-Mellon University. *ACM SIGCUE Outlook*, Spring/Summer, 46-48.

AY 83-84	AY 84-85
	Personal Computing <-- *new topic* Macintosh MacWrite & MacPaint IBM PC DOS; Epsilon / Scribble; Kermit
Mainframe Computing: TOPS 20 Mail / BBoards; Emacs / Scribe; Minitab; Help, Spell, Photo	Mainframe Computing: TOPS 20 Mail / BBoards; Emacs / Scribe; Minitab; Help, Spell, Photo
(order was not as straightforward as the above outline implies...)	
Lectures: two hours, every week six weeks	Lectures: two hours, every week seven weeks
Assignments: suggested, collected, not graded to be done outside of class in regular public clusters	Assignments: required, weekly, graded undergrad TAs scheduled to help students in regular public clusters
Proficiency Exam: required, pass / fail	Proficiency Exam: required, pass / fail
Final Grade: pass / fail proficiency exam 100% of grade	Final Grade: pass / fail proficiency exam 50% of grade assignments 50% of grade

Table 1: Computer Utilization

AY 85-86	AY 86-87	AY 87-88
Personal Computing Macintosh - 2 weeks MacWrite & MacPaint IBM PC - 2 weeks DOS; Epsilon / Scribble; Lotus 1-2-3; Kermit	Personal Computing Macintosh - 3 weeks MacWrite & MacPaint IBM PC - 3 weeks DOS; Epsilon / Scribble; Lotus 1-2-3; Kermit	Personal Computing - 6 weeks Macintosh MacWrite & MacPaint IBM PC DOS; Epsilon / Scribble; Lotus 1-2-3; PC Messages / Server
Mainframe Computing TOPS 20 - 2 weeks Mail / BBoards; Emacs / Scribe; Help, Spell, Photo	Mainframe Computing TOPS 20 - 3 weeks Mail / BBoards; Emacs / Scribe; Help, Photo	Mainframe Computing <-- *dropped!*
	Distributed Computing <-- *new!* Andrew - 3 weeks Operating System Features (windows, Unix..., help) Mail / BBoards Document Production EditText	Distributed Computing - 6 weeks Andrew Operating System Features (windows, Unix..., help) Mail / BBoards Document Production EZ and Emacs / Scribe Library Information System cT Programming Language
Each "session" ran 7 weeks, 3.0 hours / student / week	Course extended to 13 weeks, 1.5 hours / student / week	Course remained at 13 weeks, 1.5 hours / student / week
Format: Lecture (1.5 hours) 2 groups of <300 students Hands-on Lab Session (1.5 hrs) 30 groups of ≤ 40 students	Format: Slide-tape show (~ 20 min) in the same room with Hands-on practice (~ 1 hour) 70 groups of ≤ 20 students	Format: Slide-tape show (~ 20 min) then exit to adjoining room for Hands-on practice (~ 1 hour) 70 groups of ≤ 20 students
Within each half-semester: Week 1 Intro, Macintosh Weeks 2,3 IBM PC Weeks 4-6 Tops-20 Week 7 Andrew (demonstration only)	Four Quadrants Week 1 Intro, Course Structure, etc. Weeks 2-7 Group A: Andrew & Macintosh Group B: TOPS-20 & IBM PC Weeks 8-13 Group A: TOPS-20 & IBM PC Group B: Andrew & Macintosh	Two Halves Week 1 Intro, Course Structure, etc. Weeks 2-7 Group A: Andrew Group B: Macintosh & IBM PC Weeks 8-13 Group A: Macintosh & IBM PC Group B: Andrew
One Facility: 40 Macintoshes / MacCharlies used as Macintoshes able to simulate IBM PCs & as terminals to TOPS-20s unavailable as public cluster for the entire semester	Three Separate Facilities: 1. Andrew Workstations (20) 2. Macintoshes (20) 3. IBM PCs (20) - used as PCs & as terminals to TOPS-20s unavailable as public clusters during the semester	New Facility Dedicated to CSW: 50-seat lecture room two 20-seat laboratories 1. Andrew (20) 2. Macintosh & IBM PC (20 ea.) available as public cluster: fall evenings, spring and summer full time
Standard Documentation, as available in the clusters; Newsletters, weekly	"The Book"; Supplementary handouts, as needed; Jargon Sheets, weekly	"The Book" - revised; Suppl. Handouts, as needed; Jargon Sheets, weekly
Assignments: weekly for six weeks graded (six in all)	Assignments: weekly two per "quadrant" graded (eight in all)	Assignments: weekly five per "half" graded (ten in all)
Exams: could be taken if an assignment was missed	Exams: four exams at end of each "quadrant"	Exams: two exams at end of each "half"
Grades: Pass / Fail 95% Pass 5% Fail	Grades: Letter Grades 72% A 15% B 3% D 6% C 4% Fail	Grades: Letter Grades 65% A 20% B 4% D 8% C 3% Fail

Table 2: Computing Skills Workshop

SUBJECT SEARCHING IN A UNIVERSITY ON-LINE CATALOG: IMPLICATIONS FOR SCHOOL LIBRARIES

Sally E. Doyen
Daniel D. Wheeler
College of Education
University of Cincinnati

Abstract

Fourteen college students participated in a study ofsubject-searching performance in a university on-line librarycatalog. Three features of the system's menu-driven interfacewere evaluated: on-line subject headings, command windows, andon-line help. The students performed four subject-searchingtasks each of which included two or three questions. On the total set of ten questions, they scored a mean of 5.6 correct answers. Analysis of the detailed interaction logs showed that they had difficulties with the on-line subject headings list. They were successful with the command windows but did not use the help system extensively. Implications for the use of on-line catalogs in elementary and secondary schools are discussed in light of the difficulties exhibited by college students.

Introduction

The integration of technology into the curriculum has become a major theme of the educational community in recent years. One technological application that will affect both students and teachers in gathering and using information for the classroom is the computerization of library/media functions. The past decade has seen a virtual explosion in the use of computers in library settings. A recent survey of academic libraries in the United States (Camp, Agnew, Landram, Richards, & Shelton, 1987) found that 77% have or plan to implement a computerized catalog. Other writers estimate that within a few years most medium and large libraries in North America will be automated (Rochell, 1987).

Elementary and high school libraries will also be affected by this automation trend. The increased storage capacity and speed of micro- and mini-computer systems are providing individual buildings and districts the means of automating their own library holdings. By the beginning of 1986 there were about a dozen microcomputer-based catalog systems available to schools at prices comparable to that of setting up a new conventional card catalog (Costa & Costa, 1986). The recent move into the school market by companies such as Follett and IBM promises that library automation will continue to filter down from academic to school libraries. In addition, dial-access via personal computers and modems is making university library databases available to school libraries and media centers.

On-line database vendors such as Dialog and Dow-Jones have recognized as well the market potential that currently exists in schools and have responded by offering them special educational discounts. As of April, 1987, over 800 high schools had subscribed to Dialog's educational program for on-line searching (Schill, 1987). This number will no doubt continue to increase as awareness of the potential for information-access through telecommunications grows.

Research into on-line catalog usage in academic libraries has indicated that users prefer the on-line catalog to the card catalog, but nevertheless experience difficulty in certain aspects of on-line searching, especially searching by subject (Matthews, Lawrence, & Ferguson, 1983; Markey, 1985). Despite this difficulty, subject searching has increased dramatically in on-line catalogs and has replaced author and title searching as the predominant form of catalog access (Steinberg & Metz, 1984; Alzofon & Van Pulis, 1984). The occurrence of subject searching decreases, however, as users progress academically, with undergraduate students performing more subject searches than faculty (Frost, 1985).

Librarians have long recognized the difficulty of subject searching, a process that requires the user to translate an often-vague concept into the controlled vocabulary of the catalog (Cochrane, 1985). Bates (1986) has suggested that a user's success may hinge on the ability to generate multiple concepts to pursue as searching terms.

Such a suggestion, however, is virtually meaningless when directed toward elementary school children, who are struggling with basic vocabulary development and who do not possess the cognitive ability or knowledge base to generate synonyms or expand their ideas into more abstract concepts. To a fourth grader, Mom or Dad drives to work in a car, not an automobile, and Jack and Jill is a magazine, not a periodical.

Younger students even have difficulty discriminating between subject cards and title cards in the traditional card catalog. Because of this difficulty, school libraries often have divided card catalogs with authors and titles in one catalog and subjects in another. However, students still struggle with this basic bibliographic distinction. Unlike card catalogs, on-line catalogs cannot be "divided" in such a way as to provide a visual referent to teach students the distinction among types of access points into the library's collection. How students' conceptual models of the computerized database and its components will be shaped is unknown.

Interest in how conceptual models develop in learners has generated recent research in human-computer interaction, although this research has focussed primarily on adults. It indicates that users form their own conceptual frameworks, or mental models, of the structure and internal processes of the physical systems with which they interact (Norman, 1983). These models are shaped by the type of system interface the user encounters as well as by the user's prior knowledge and experience with the system. The assumption held by researchers is that this user model can be influenced by both instruction and computer interface design (Norman, 1983).

On-line catalog users' mental models are shaped by experience with the computer system, and by existing knowledge in domains that form the on-line searching environment: namely, computer experience and card catalog experience. Elementary and secondary school students with little experience in either domain will be even more dependent on instruction and interface features designed for their level of comprehension.

A common approach to simplifying the learning process for novice users of computer systems is the development of menu-driven interfaces. Research has indicated that novices both prefer and perform better with menus rather than with complete command-driven systems (Wallace, Anderson, & Shneiderman, 1987; Parton, Huffman, Pridgen, Norman, & Shneiderman, 1987). On-line catalogs usually offer a hybrid of menu- and command-driven interfaces. For example, users may choose from a list the option of author, title, or subject searching, but then must type in a specific entry, a manual skill that would be difficult for many young students.

As a further aid to novice searchers, on-line catalogs often provide help features accessible to the user during the searching process. Unfortunately, research has shown that users have difficulty with on-line help (Baker, 1986) and will switch to off-line assistance when they cannot interpret screen messages (Janosky, Smith, & Hildreth, 1986). Users with weak reading skills are likely to panic when confronted with one or several screens of printed information that is supposed to help them correct an error made in searching. Findings, therefore, that adult users do not read on-line documentation (Wright, 1984) have serious implications when considering younger students who have less experience upon which to draw for self-help.

This paper presents findings from a research study conducted with college students using an on-line catalog. The purpose was to investigate how users of a menu-driven system pursued subject-searching tasks. Three specific features designed to simplify the searching process were tested: (a) an on-line subject headings list, (b) pop-up windows that contained commands, and (c) an on-line help system.

Method

Subjects

Fourteen students (9 females, 5 males) were recruited from undergraduate and graduate education classes. None of the students had previously seen the new interface to the on-line catalog that included the features outlined above, although 13 of the 14 students had used an earlier version of the system.

On-line System

The on-line library catalog used in this study was located at a major Midwestern university, where it had been available for two years. The prototype system evaluated in this study included three major changes, access to an on-line subject headings list, the addition of command windows, and an expanded help system.

The on-line subject headings file was the first major new feature of the system, accessed through a "subject browse" command. Other than a brief statement to "Use subject browse if unsure of subject," no other instruction was provided.

The earlier versions of the system displayed a menu of options whenever the user needed to make a choice. The continuous menu display added to the response time and limited the amount of the screen that could be used to display the information retrieved by the system. The revised version used the full screen for information display and showed the commands in a three-window sequence accessed by pressing the enter key. An overview of the 3-windows was not provided by the system. At the beginning of the session, students did not know how many windows existed, nor their contents.

The help system was also expanded. Students could access a help screen via a menu option after retrieving either too many or zero items. They could also enter a question mark to get to a help screen from any other screen.

For this study, a microcomputer connected to the on-line catalog through a modem was used to record each searching session. It kept a complete log of interactions with the system. Tasks

Students were given four written searching tasks, such as finding books on the topic of "boating accidents." The two questions for this task requested an author's name and call number for a book on the topic. There were two or three short- answer questions for each of the remaining tasks, for a total of ten questions.

All four tasks were designed to be approached as subject searches, but only the first offered no other searching options (such as author or title) within the question. The remaining tasks provided some author or title information that could be used for the search, although each included a subject term with which users could begin.

Procedure

Each student was tested individually. They reported to a library training room and were seated at a microcomputer workstation. After students completed a brief questionnaire requesting a report of prior experience with the on-line catalog and basic student demographic information, the investigator explained how to use the keyboard. Students were then instructed to pursue the four searches in the order in which they appeared on the worksheet. They could end a task when they felt they had found the answers or when they did not know how to pursue further. Students were told to pursue the tasks as if they were in the library and no one was available to help them. The investigator did not answer questions during the tasks but remained in the room to assist with technical problems (e.g., the keyboard locking).

When students completed the four tasks to their satisfaction, the investigator reviewed with them the problems they had during their searches and demonstrated ways in which they could have improved their searches. Students were asked to explain what the on-line subject headings list was and the problems they had with it. Students who had had experience with the previous interface of the on-line catalog were asked which of the systems they preferred. Each session lasted approximately one hour.

Results and Discussion

Out of the 10 questions on the searching tasks, students averaged 5.6 items correct (SD = 1.4) with a range of 3-8 correct. The mean time for the tasks was 19.4 minutes (SD = 5.7), with a range of 13.7-33.9 minutes.

This level of performance clearly indicates that students have problems carrying out subject searches using the on-line catalog. The finding that twelve of the thirteen students who had used the previous menu-system preferred the new system corroborates earlier findings that users like computerized catalogs even if they are not completely successful in using it (Matthews, Lawrence, & Ferguson, 1983).

Each of the new features being investigated is discussed separately in the following sections.

Subject Searching and the Use of Subject Browse

Although all four searching tasks were designed as subject searches, the use of a subject approach dwindled over the course of each session. All 14 of the participants used a subject approach for the first task, 9 used a subject approach for the second, and 3 used a subject approach for each of the two remaining tasks. Students seemed to prefer searching by author or title when that information was included as part of the task information.

Subject searches were generally unsuccessful because participants did not use the appropriate subject term. Twelve of the 14 students accessed the subject headings list during either question one or two, following an unsuccessful subject search attempt. Across all students, there were seventeen attempts to use the subject headings list to locate a subject term. Only seven of these attempts were successful. In the other ten cases the students left the subject headings list without having found a correct subject term. Six of eight students who encountered a screen message explaining how to modify an incorrect entry failed to make the necessary correction when they returned to their searches.

Some students were even confused as to what the subject headings list contained. Three people thought the display was a list of books, and three thought the first screen contained everything related to their topic, failing to realize that additional subject terms might appear on subsequent screens.

Although most students seemed to understand that the subject browse feature could provide correct subject terms with which to begin a search, they accessed the list only after having failed in a subject search. The brief explanation ("Use a subject browse if unsure of terms") was adequate to get people into this feature. However, once there, many students did not use it effectively. The exclusive use of this feature at the beginning of the session (during questions 1 and 2 only) may indicate that users did not find it helpful enough to return to it.

Use of windows

The purpose of command windows in an on-line catalog is that the full screen can be used to display information that has been retrieved by the user. The window is overlaid on the display, so that the user can see the commands while maintaining access to the displayed information. In the system being evaluated in this study, the window is not displayed until the user presses the "enter" key. Following some initial hesitation, participants had no trouble with this method of accessing the windows in order to select appropriate commands.

Commands can also be entered directly without requesting the command window first, although this capability is never explained to users. Instead, they must infer it as they interact with the system. Eleven participants began direct entry of a command at some point during the session. The other three either did not realize that they could skip the menu display step, or chose not to learn the commands they needed.

Only three participants viewed the entire 3-window sequence at some point during the session. Therefore, it seems crucial that the most commonly used commands be put in the first window.

Use of optional help screens

Five students accessed the optional help screens a total of 7 times during the session. Six of the 7 uses were for "zero-items-retrieved" search results, while one was for "too-many-authors" help. One student tried twice to apply the examples on the screen, was unsuccessful both times, and did not return to the help screens during the session. The remaining students who accessed these screens appeared to read the screens, but did not apply the examples or suggestions in their next entries. In fact, two students repeated the exact entry that had already generated a "zero-items-retrieved" response.

Minimal use of optional help systems has been found in other research as well (Wright, 1984). Apparently, users prefer to try self-generated alternatives rather than to access and read on-line documentation. They may also fail to understand what help means when they see it as a menu option and consider it a diversion from their goal of finding the answer to a searching task. In this study, help screens were accessed as a response to receiving no items, rather than too many. Students obviously do not consider having obtained too much information to be a problem. Students' willingness to accept computer responses that failed to answer the questions also indicates a belief that any information is better than none.

Conclusions

The problems exhibited by this study's participants can be best understood in terms of the inadequacy of their mental models. In order to use the on-line catalog effectively, users must have a mental model that encompasses the organization of bibliographic information as well as operation of the on-line catalog. The lack of an adequate model made it impossible for many students to understand and apply the information they retrieved in the subject browse feature and the help system.

Menu-driven systems are generally regarded as easy for novices to use (Wallace, Anderson, & Shneiderman, 1987). Given the past difficulties users have exhibited in subject-searching (Markey, 1985), augmenting an existing menu-driven interface with the features outlined in this study is a logical step in an on-line catalog's maturation. The findings of this study, however, suggest that merely providing additional options through a menu system does not guarantee searching success. The difficulty in using a subject headings list is not ameliorated by putting it on line. Tools which should theoretically improve the searching process will be effective only if users possess a good conceptual understanding (mental model) of the system and task. Assisting in the development of this model through instruction may be a better approach than trying to simplify the system too much in the hope that it can be operated without instruction.

Subject searching is even more important at the elementary and high school levels than it is at the college level. Younger students are rarely looking for a specific book title or specific author. They want information about some topic. Given the difficulties experienced by college students, it is safe to assume that younger students will exhibit even greater problems in subject searching. The challenge lies in designing what Van der Veer, Tauber, Waern, & Van Muylwijk (1985) call metacommunication, including both on-line and off-line documentation, that will contribute to effective use of the sytstem.

As young students interact more and more with computers, and particularly on-line catalogs, their mental models will develop. If those models are to be accurate and useful, instruction and interfaces must convey clearly the essential information needed to perform given tasks and to learn the principles of the system which can be transferred to other computer applications.

How this instruction, whether off-line or on-line, can best be presented is the challenge. Much of library user-education has used the card catalog as the logical comparison to the on-line catalog (Borgman, 1986). Students without the printed referents of catalog cards and subject-headings lists will need conceptual frameworks based on other areas of familiarity.

Designers of these systems must be cognizant of the needs of younger users, even more so than with adult populations. The wide disparity of abilities and skills between elementary and high school students calls for an interface adaptable to many levels. Care must be taken not to direct the systems toward the librarians who already understand the concepts of bibliographic access, Boolean logic, and subject-heading hierarchies. The systems must be evaluated during development so that help features, screen messages, and searching options follow a coherent model and do not mislead or confuse. Observing students as they pursue tasks, asking questions, and recording searches for subsequent analysis are research techniques that can be used with younger students. Our work with college students only suggests the kinds of trouble younger students may encounter. Much research with younger students will be needed to develop systems that they can use as effective access tools in school libraries.

References

Alzofon, S. & Van Pulis, N. (1984). Patterns of searching and success rates in an online public access catalog. *College & Research Libraries*. 45, 110-115.

Baker, B. (1986). A conceptual framework for teaching online catalog use. *Journal of Academic Librarianship. 12*, 90-96.

Bates, M. (1986). Subject access in online catalogs: a design model. *Journal of the American Society for Information Science. 37*, 357-376.

Borgman, C. (1986). The user's mental model of an information retrieval system: an experiment on a prototype online catalog. *International Journal of Man-Machine Studies. 24*, 47-64.

Camp, J., Agnew, B., Landram, C., Richards, J. & Shelton, J. (1987). Survey of online systems in U.S. academic libraries. *College & Research Libraries. 48*, 339-350.

Costa, B., & Costa, M. (1986). *A micro handbook for small libraries and media centers*. Littleton, CO: Libraries Unlimited.

Cochrane, P. (1985). *Redesign of catalogs and indexes for improved online subject access: selected papers of Pauline A. Cochrane*. Phoenix, Arizona: Oryx Press.

Frost, C. (1985). *Student and faculty subject searching in a university online public access catalog*. (ERIC Document Reproduction Service No. ED 264 872)

Janosky, B., Smith, P., & Hildreth, C. (1986). Online library catalog systems: an analysis of user errors. *International Journal of Man-Machine Studies*. *25*, 573-592.

Markey, K. (1985, January/March). Subject-searching experiences and needs of online catalog users: implications for library classification. *Library Resources & Technical Services*. pp. 34-51.

Matthews, J. Lawrence, G., & Ferguson, D. (1983). *Using online catalogs: a report of a study sponsored by the Council on Library Resources*. New York: Neal-Schuman.

Norman, D. (1983). Some observations on mental models. In Gentner, D. and Stevens, A. (Eds.) *Mental models*. Hillsdale, New Jersey: Lawrence Erlbaum, pp. 7-14.

Parton, D., Huffman, K., Pridgen, P., Norman, K. & Shneiderman, B. (1985). Learning a menu selection tree: training methods compared. *Behavior and Information Technology*. *4*, 81-91.

Rochell, C. (1987). The next decade: distributed access to information. *Library Journal*. *112* (2), pp. 42-48.

Shill, H. (1987). Bibliographic instruction: planning for the electronic information environment. *College & Research Libraries*. *48*, 433-453.

Steinberg, D. & Metz, P. (1984). User response to and knowledge about an online catalog. *College & Research Libraries*. *45*, 66-70.

Van der Veer, G., Tauber, M., Waern, Y., & Van Muylwijk, B. (1985). On the interaction between system and user characteristics. *Behavior and Information Technology*. *4*, 289-308.

Wallace, D., Anderson, N. & Shneiderman, B. (1987). Time stress effects on two menu selection systems. In *Proceedings of the Human Factors Society 31st Annual Meeting*. Santa Monica: Human Factors Society.

Wright, P. (1984). Manual dexterity: a user-oriented approach to creating computer documentation. In Janda, A. (ed.) *Human Factors in Computing Systems: Proceedings of the CHI '83 Conference*. Amsterdam: Elsevier.

RETHINKING LOGO PEDAGOGY USING A PROCESS APPROACH

Daniel H. Watt and Molly L. Watt
Senior Associates
Education Development Center
Center for Learning Technology
55 Chapel Street
Newton Massachusetts 02160
(617) 969-7100

Abstract

This paper suggests a process approach to Logo pedagogy based on observations of Logo programming projects carried out by adults and children. We describe some limitations in the classroom practices of Logo teachers, and suggest remedies modelled on the successful implementation of a process approach to writing instruction.

The development of these ideas was supported in part by the National Science Foundation as part of *Exploratory Research on Critical Aspects of Logo Learning: A Collaborative Project With Teachers as Observers*, grant number MDR 865 1600. We would also like to acknowledge the contributions of research consultants, Roy Pea of New York University and Jan Hawkins of Bank Street College of Education; our Research Assistant, Joan Funk and Administrative Assistant, Gerry Sills of Education Development Center; Advisory Committee members, Elizabeth Bjork, Douglas Clements, Claryce Evans, Judith Kull, Beth Lowd, Paul Goldenberg and Sylvia Weir; and collaborating teacher researchers, Elizabeth Bermer, David Crump, Diana Freedman, Doreen Kelly, Robert Kondel, Jane Manzelli, Mary Miller-Teehan, Pasquale Puleo and Kitsy Rothermel. Any ideas, opinions, findings and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation or of any of our collaborators.

Introduction

A research process inevitably starts with questions, but it doesn't always end with answers. In fact, we prefer to think of research as a cyclical process in which general hypotheses lead to specific questions, questions lead to data gathering, data gathering leads to analysis, and analysis leads to new questions and new hypotheses. In some cases, the cycle proceeds in an orderly way, with new hypotheses and questions developing smoothly out of research findings, and leading to a gradually improved understanding of the circumstances under investigation. In other cases, unanticipated observations may redirect the research into a new directions, leading to hypotheses and questions that were not envisioned by the researchers in the first place.

In this paper, we will share some unanticipated observations and hypotheses that arose during a research project entitled *Exploratory Research on Critical Aspects of Logo Learning: A Collaborative Project with Teachers as Observers* (NSF Grant # MDR 8651600). Our project began with the goal of finding ways for teachers to work as researchers in their own classrooms to assess the Logo learning of their students. In the process, we incidentally uncovered what we believe to be some general weaknesses in the teaching approaches used by many Logo teachers. And we began to formulate ideas for a new approach to Logo pedagogy.

The project was established in response to several widely recognized educational needs:

- Logo has not yet lived up to the expectations engendered by Seymour Papert's influential book, *Mindstorms* (Papert, 1980). There is a need for better information about what Logo students are actually learning in classrooms (Watt and Watt, 1986a; Papert, 1987; Clements, 1985; Pea, 1987; Becker, 1987; Walker, 1987).
- Teachers who teach Logo in schools need better understandings of the Logo content they are teaching, and of the learning strategies they are attempting to foster through Logo experiences (Watt and Watt, 1986b).
- In order to fulfill themselves as professionals as well as to contribute to improving the educational experiences of their students, teachers need ongoing opportunities for professional development (Carnegie, 1987, Holmes 1987, Lambert 1987, Loucks-Horsley, et. al. 1987).

In response to these needs we designed a project in which a group of experienced Logo teachers engaged in collaborative classroom research to investigate the Logo learning of their own students (Watt and Watt, 1986a). The project had two intertwined components: the first involved supporting the professional development of Logo teachers by working with them, over a six month period, in a collaborative classroom research process. The major activity of this component of the project was a series of eleven research seminar meetings for all project participants and staff. We reported on this aspect of the work in last year's NECC proceedings (Watt and Watt 1988), and describe it more fully in a final report available from Education Development Center (Watt and Watt, 1988).

The second component was the investigation which provided the focus for the collaborative research: What were Logo students learning in particular classrooms, and how could that learning be enhanced? For this component we developed a list of critical aspects of Logo learning -- aspects of Logo that can add to a learner's power and knowledge if they are used and understood by the learner, and that usually require some type of timely intervention by a teacher in order to be learned effectively (Watt and Watt, 1986a). This list was developed and refined in collaboration with classroom teachers, as they conducted research observations in their own classrooms. This part of our work is summarized in a series of articles published in *The Logo Exchange* (Watt, D. 1988-89).

Limitations of Logo pedagogy and Logo classroom contexts

We intended to have our work focus on the *content* of Logo learning, and on the knowledge needed by teachers to help children learn that content most effectively. However, we soon found that we had to pay attention to classroom *contexts* as well. Our collaborators in the project were twelve classroom teachers and computer coordinators from four Boston area school districts. All four districts included Logo activities as a required curriculum component for elementary students. All of our collaborative researchers were experienced, successful teachers with at least two years experience teaching Logo. However, although circumstances and practices varied among the classrooms involved in our research project, we found the following to be true in most cases, as soon as we began to visit classrooms:

- Neither teachers nor students kept regular records of student work. Printouts were rarely used for reference by teachers or students, and few examples of student Logo work were on display.

- A common teaching strategy was to present general challenges to an entire class, and have students demonstrate their results to each other when they were finished.

- Students were rarely asked to revise or reflect on their work except to correct obvious errors. Once a student created a working draft, both students and teachers considered it finished.

- Teachers would introduce new concepts to an entire class, but would have little direct interaction with students while they were working, unless the student specifically asked for help.

- In general, we would have to say that although Logo activities were part of the required curriculum in each of the four school districts, Logo was not taught or evaluated as thoroughly as other subjects. Neither students nor teachers were held accountable for Logo learning in three of the school districts. In the fourth district, a *minimal competency* in Logo (as measured by a criterion-referenced test) is required for all students completing grade 5.

Concerns about these classroom context issues about Logo learning led us to consider the general pedagogical approaches being used for Logo instruction, and to a search for classroom support structures and teaching interventions that could maximize Logo learning, while maintaining Logo's exploratory spirit and learner-centered values.

Logo was developed in the late 1960s and early 1970s under the leadership of Seymour Papert at MIT. Papert and his colleagues created Logo to be a learning environment that would allow children "... to *do* mathematics, rather than merely to learn *about* it (Papert, 1971)." Drawing on his years of work with psychologist Jean Piaget, Papert and his colleagues set out to create a computer-based learning environment and a pedagogical approach that were based on ways that children construct their own learning naturally. In his influential book, *Mindstorms*, Papert called this approach Piagetian learning, which he described as "learning without curriculum" and "the kind of learning that happens without deliberate teaching (Papert, 1980)."

Papert went on to explain that "'Teaching without curriculum' does not mean spontaneous, free-form classrooms, or simply 'leaving the child alone.' It means supporting children as they build their own knowledge with materials drawn from the surrounding culture (Papert 1980)." However, he never emphasized what a Logo teacher should know or particular teaching strategies to be used. Many educators developed the expectation from reading *Mindstorms* that a Logo-speaking computer, plus knowledge of a few Logo commands were all that was needed for a child to learn Logo and to acquire powerful ideas from computer science and mathematics.

Although since 1980 there has been much criticism of Papert's approach (see for example, Moursund, 1983, Pea and Kurland, 1984; Kurland and Pea, 1985), few research studies have focussed on classroom strategies for teaching Logo. An Ontario study (Carmichael, et. al, 1986) with beginning Logo teachers described some of the demands that Logo teaching places on a teacher, and suggested some classroom arrangements and schedules that might facilitate Logo learning. The final report of that study concluded that the most effective learning environments for Logo are those in which teachers provide knowledgeable support for their students and that in order to do this effectively, teachers need support themselves. Hawkins' (1985) study of two beginning Logo teachers showed their shift from a discovery learning approach to a more structured approach which included direct teaching of specific concepts and skills. A study at Vanderbuilt University demonstrated that a group of students who were given structured tasks to perform during Logo classes were found to have better mastery of specific Logo concepts than students who were allowed to explore Logo freely (Delclos, et. al., 1985, Kinzer, et. al., 1985).

None of these studies focus directly on the classroom contexts and teacher knowledge needed to teach Logo effectively. Although some educators have developed Logo curricula that substitute structured teaching sequences for exploration and student-generated projects, in an attempt to ensure that students will learn a particular set of Logo commands and skills, we believe that the distinction between free exploration and structured classroom instruction provides an insufficient conceptualization for organizing Logo teaching.

In order to teach Logo effectively we believe that a teacher must be able to assess student learning, and know how to guide students toward more powerful knowledge. Our objective is to create classroom environments which supports students in learning specific skills and concepts while engaged in creative, generative projects of their own choosing. This suggests to us that a Logo teacher should understand the processes by which Logo projects develop and reach completion. In this paper we suggest a model, based on our observations of what Logo programmers actually do, and on the successful implementation, over a period of years of a process approach to writing instruction (Watt and Watt, 1987c).

A Logo Programming Process

Our suggested approach to Logo pedagogy begins with observations about what Logo programmers, adults as well as children, actually do in initiating and completing typical Logo projects. In watching many Logo programmers create their own Logo projects, we have identified the following working modes, through which a Logo project develops from conception to completion. These include:

- *brainstorming and explorations*: generating ideas, testing hypotheses, extending earlier work (one's own and others);

- *choosing a project*: narrowing the domain, selecting goals, drawing a picture or describing desired outcomes in general terms;

- *planning and simplifying the project*: organizing the work to be done, describing outcomes more specifically, labelling a picture with specific names for parts of a drawing, making a procedure tree, dividing the work into parts, revising general objectives to select the easiest subgoals to accomplish first;

- *creating a working draft*: getting the pieces built, put together and working; at this point the result may be somewhat ragged, not as pretty as the original intention, and incomplete;

- *debugging, revising and elaborating*: fixing things that don't work, reformatting text and images on the screen, rescaling sizes, adding color and other elaborations, eliminating interactions that allow a program to crash; sometimes a program is significantly modified or elaborated at this point, so that a programmer has to return to the mode of creating a working draft for the newly added parts of the project;

- *sharing through publication*: revising with particular users in mind, fine tuning both the output, and the text of a program, polishing, changing subprocedure names, shortening program lines, making everything as readable as possible.

The process of developing a Logo project is usually recursive, rather than linear (see Figure 1a and Figure 1b). As shown in Figure 1b, a programmer is likely to move back and forth flexibly among different modes while working on any particular project.

Looking for comparable models: the Writing Process Approach

The process we have described has much in common with learning experiences in other domains that include generative, problem-solving processes in which students set their own goals, work on problems of interest to them, and express their own interests through their work. Among these are the processes of scientific inquiry (Hawkins, Mawby and Ghitman, 1987, Hawkins and Pea, 1987), Polya's approach to solving mathematical problems (Polya, 1957, Schoenfeld, 1987), and the writing process described by Graves (1983), Daiute (1985), Scardamalia and Bereiter (1986), and Pea and Kurland (1984, in press), among others. Each involves distinct modes such as exploration and brainstorming, planning, applying plans, reflecting on partial results, revising plans or practices in response to reflection, and sharing work with appropriate audiences.

In our review of educational research literature, both theoretical and practical, we find that reforms in writing instruction most nearly approximate the changes we are suggesting for Logo pedagogy. Research on classroom contexts for supporting writing was based on research into the practices of successful writers. This research has led to new practices in teaching writing and training writing teachers, that have been adopted successfully in many parts of the country (Scardamalia and Bereiter, 1986, Graves, 1983, and Bay Area Writing Project, 1983). We believe that some of the specific instructional strategies and teacher development strategies used by writing process advocates may be useful models for resolving some of the difficulties we have observed in Logo classrooms. For example:

1. Writing process teachers recognize that young writers learn to write most effectively by writing about subjects of their own choice (Graves, 1983). Many Logo teachers would probably agree: they consider Logo to be a creative activity, and allow children to spend at least part of their Logo work time on activities of their own choosing.

2. Writing process teachers recognize that there are distinct modes of writing which require different types of instructional support or scaffolding (Graves, 1983; Scardamalia, Bereiter and Steinbach, 1984; Scardamalia and Bereiter, 1986). Graves (1983) stresses the use of teacher/student conferences to provide differentiated support for different modes of the writing process while Scardamalia and Bereiter (1986), and Morocco and her

colleagues at EDC have developed teaching approaches based on procedural facilitation, for example, teaching children specific procedures for planning or revising a piece of writing (Morocco, 1987; Morocco and Neuman, 1985, Morocco, et. al. 1987).

However, it is our observation that most Logo teachers do not make such distinctions between working modes; nor do they consciously provide different types of support for different working modes. This has not received a great deal of attention in the Logo research literature either.

3. Writing process teachers recognize that students (like successful adult writers) write more effectively when they write for real audiences. Therefore they create opportunities for their students to get feedback from peers about their work in progress, and the opportunity to publish some of their work for specific audiences (Applebee, 1981; Cohen and Miyake, 1985; Cohen and Riel, 1986; Freedman, 1985; Graves, 1983; Hillocks, 1986; Scardamalia and Bereiter, 1986).

At present, most Logo student work is considered finished at a stage that we would consider a "working draft." It is our conjecture that Logo students who have specific audiences in mind for their work may be motivated to revise their work. In the process we suspect that they might learn some of the important concepts which they may be missing in their present Logo learning experiences. The notion of audience for student work is conspicuously lacking in Logo research literature (Watt and Watt, 1987a, 1987b).

4. Finally, we observe that successful professional development practices used to foster the writing process approach usually include two key elements (Bay Area Writing Project, 1983). First, they provide opportunities for teachers to be writers themselves and experience all phases of the writing process. Second, they attempt to share effective teaching strategies for supporting different modes of the writing process. This suggests to us that Logo teachers should have some experiences with programming projects at their own level, rather than merely learning the content their students are expected to learn, as well as opportunities to benefit from whatever is known about practical methods to support students in the different modes of the Logo programming process.

Related work

As we have developed these ideas, we have learned about similar efforts in other places. Noss (1985), observing the use of Logo with children in England, identified three working modes:

1. making sense of a new command
2. exploration
3. goal-directed activity.

Noss noted that once his students had mastered the first mode and understood how to use a feature of the language, they would switch back and forth between the second and third modes. He also observed that his students could learn and understand a new concept more effectively if it was introduced to them during an exploratory activity, rather than during a goal-directed activity. If they needed help in completing a specific goal-directed project, they might use a new idea and get on with their work, but would not take the time to fully understand the new concept.

Thyberg (1987) has clearly articulated the ways that she used the writing process approach to support both Logo and writing in her multi-grade primary classroom at the University of Pittsburgh Lab School. Freyd (1987) at Pennsylvania University has turned the model around, using students' Logo experiences as a basis for enhancing their language learning. And the INQUIRE project at Bank Street College is developing software designed to support a recursive or cyclical inquiry process approach to the learning of science in middle school grades (Hawkins, Mawby and Ghitman, 1987, Hawkins and Pea, 1987).

What needs to be done next?

The ideas we have set forth in this paper are not yet sufficiently developed to serve as a workable guideline for Logo teachers. But they do suggest a research and development agenda that could inform Logo-using educators and eventually lead to an effective, widely-used Logo pedagogy based on a process approach. Several issues and questions relating to revision of Logo work by students, creation of audience for students' Logo projects, and appropriate support structures for different modes of the Logo process, seem to us to offer especially fruitful next steps.

How important is revision in extending and enhancing student learning? What kinds of revisions do students commonly make in their Logo projects? How does revision of Logo projects effect the learning of important ideas in problem-solving, computer science and mathematical thinking? Will an emphasis on creating programming projects for specific audiences motivate students to revise their work more regularly and more thoughtfully? Fortunately, the existing body of research on revision in writing can serve as a guide for identifying specific research questions and methods (Fitzgerald, 1987).

During the course of a school year, writing teachers may create literally dozens of publishing opportunities for their students. But what are appropriate audiences and publishing vehicles for Logo programming projects? Here we simply need to gather lots of ideas and try them out in classrooms. Some initial suggestions for creating audiences for student Logo projects include: publishing books of Logo projects that combine both graphics and procedures; sharing Logo disks and documentation among different groups of students, perhaps via telecommunication; printing Logo designs and procedures on T-shirts and bulletin boards; creating opportunities for Logo programmers to *use* all or part of each other's work in their own programs; collaborative programming projects involving several students; and developing educational games and teaching software for use by other students.

Many of the above ideas are already used by some teachers. What's needed, however, is to incorporate them in a clearly articulated process approach. This implies developing specific support structures for different modes of the process. Following the work of Graves on writing (1983) we suggest the use of student-teacher conferences, with the focus of the conference changing for different modes of the process (see Figure 2).

Another practice derived from the work of good writing teachers might be a programmer's circle, similar to an author's circle in the writing process (see, for example, Boles, 1985). Thyberg (1987) uses an editorial board of students whose task is to review and make suggestions about student work before it can be publicly shared with the rest of the class.

We are confident that the approach suggested here is a practical one which can be used effectively by many Logo teachers. Success with a writing process approach by many teachers at all teaching levels lends credence to this view. We observe that many Logo teachers already use a writing process approach when they teach language arts. Such teachers should be able to readily adapt their existing practices to Logo instruction. Although there may be many other valid ways to teach Logo, we believe that this approach is in keeping with the intentions of Logo's original developers to create a learning environment in which children construct their own knowledge, and learn important skills and concepts of mathematics, problem-solving and computer science, while working on open-ended projects that are meaningful to them and that serve their own purposes.

References

Applebee, A. (1981). "Writing in the Secondary School: English and the Content Areas," *NCTE Research Report No. 21*, National Council of Teachers of English, Urbana IL.

Bay Area Writing Project (1983). "The National Writing Project Evaluation Portfolio," from National Writing Project Evaluation Studies, 1977-1982, Berkeley CA.

Becker, H.J. (1987). "The Importance of a Methodology That Maximizes Falsifiability: Its Applicability to Research about Logo," *Educational Researcher*, Vol. 16, No. 5, June/July 1987.

Boles, K. (1985). "The Writing Process," classroom videotape, Brookline Public Schools, Brookline MA.

Carmichael, H., et. al. (1986). *Computers, Children, and Classrooms: A Multisite Evaluation of the Creative Use of Microcomputers by Elementary School Children*, Ministry of Education, Ontario, Canada.

Carnegie (1986). *A Nation Prepared: Teachers for the 21st Century*, Carnegie Forum on Education and the Economy, Washington DC.

Clements, D. (1985). "Research on Logo in Education: Is the Turtle Slow But Steady, or Not Even in the Race?" *Computers in the Schools*, Vol. 2 Nos. 2/3, Summer/Fall 1985.

Cohen, M., and Miyake N. (1985). "A Worldwide Intercultural Network: Exploring Electronic Messaging for Instruction," *Interactive Technology Laboratory Report #8*, Center for Human Information Processing, University of California, San Diego, CA.

Cohen, M. and Riel, M. (1986). "Computer Networks: Creating Real Audiences for Students' Writing," *Interactive Technology Laboratory Report #15*, Center for Human Information Processing, University of California, San Diego, CA.

Daiute, C. (1985). *Writing and Computers*, Addison-Wesley, Reading MA.

Delclos, V. et. al. "Teaching Thinking Through Logo: The Importance of Method," *Roeper Review*, Vol. 7 No. 3.

Fitzgerald, J. (1987). "Research on Revision in Writing," *Review of Educational Research*, Winter 1987, Vol. 57, No. 4, pp 481-506.

Freedman, S. (1985). *The Acquisition of Written Language: Response and Revision*, Ablex Publishing Co., Norwood NJ.

Freyd, P. (1987). "Bugs in the Workshop: Using Logo to Enrich Language Experience," *Proceedings, National Educational Computing Conference.*

Graves, D. (1983). *Writing: Teachers and Children at Work*, Heinemann, Portsmouth NH.

Hawkins, J. (1985). "The Interpretation of Logo in Practice," *Technical Report No. 34*, Bank Street College of Education, New York, NY

Hawkins, J., Mawby R. and Ghitman, J. M. (1987). "Practices of Novices and Experts in Critical Inquiry," in Pea and Scheingold, Eds., *Mirrors of Minds: Patterns of Experience in Educational Computing*, Ablex Publishing, Norwood NJ.

Hawkins, J. and Pea. R.D. (1987). "Tools for Bridging the Cultures of Everyday and Scientific Thinking," *Journal of Research in Science Teaching*, Vol. 24, No. 4, pp. 291-307.

Hillocks, G. Jr.(1986). "The Writer's Knowledge: Theory, Research and Inplications for Practice," in Petrosky and Bartholomae, Eds., *The Teaching of Writing: 85th Yearbook of the National Society for the Study of Education, Part 2.*

Holmes Group (1986). *Tomorrow's Teachers*, The Holmes Group, East Lansing MI.

Kinzer, et. al. (1985). "Different Logo Learning Environments and Mastery: Relationships Between Engagement and Learning," *Computers in the Schools*, Vol. 2, Nos. 2/3, Summer/Fall 1985.

Kurland, D. M. and Pea, R. D. (1985). "Children's Mental Models of Recursive Logo Programs," *Journal of Educational Computing Research*, Vol. 1, pp 235-243.

Kurland, D. M., Pea, R. D., Clement, C., Mawby, R. (1985). *A Study of the Development of Programming Ability and Thinking Skills in High School Students*, Bank Street College of Education, New York NY.

Lambert, L. (1986). "The End of Staff Development," Marin County Office of Education, Pt. Richmond CA.

Loucks-Horsley, S. et. al. (1987). *Continuing to Learn, A Guidebook for Teacher Development*, National Staff Development Council, Oxford, OH.

Morocco, C. (1987). *Teachers, Children and the Magical Writing Machine: Instructional Contexts for Word Processing with Learning Disabled Children*, Final Report, The Writing Project, Education Development Center, Newton MA.

Morocco, C. and Neuman, (1985). "Word Processors and the Acquisition of Writing Strategies," The Writing Project, Education Development Center, Newton MA.

Morocco, C. et. al. (1987). "'I know what to say!' Writing Activities for the Magical Machine," The Writing Project, Education Development Center, Newton MA.

Moursund, D. (1983). "Logo Frightens Me," *The Computing Teacher*, Volume 11, Number 5.

Noss, R. (1985). *Creating A Mathematical Environment Through Programming: A Case Study of Young Children Learning Logo*, Ph.D. thesis, Centre for Science and Mathematics Education, Chelsea College, University of London, London England.

Papert, S. (1971). "Teaching Children to Be Mathematicians Vs. Teaching About Mathematics," *Logo Memo #4*, MIT Logo Group, Cambridge, MA.

Papert, S. (1980). *Mindstorms*, Basic Books, New York NY.

Papert, S. (1987). "Computer Criticism Vs. Technocentric Thinking," *Educational Researcher*, Vol. 16, No. 1, January/February 1987.

Pea, R.D. (1987). "The Aims of Software Criticism: A Reply to Professor Papert," *Educational Researcher*, Vol. 16, No. 5, June/July 1987.

Pea, R.D. and Kurland, D.M. (1984) "Logo Programming and the Development of Planning Skills," *Technical Report #16*, Bank Street College of Education, Center for Children and Technology, New York NY.

Pea, R.D. and Kurland, D.M. (in press). "Cognitive Technologies for Writing," *Review of Research in Education*, Vol. 13.

Pea,R., Kurland, D. M., and Hawkins, J. (1987). "Logo and the Development of Thinking Skills," *Mirrors of Minds: Patterns of Experience in Educational Computing*, Pea, R. and Scheingold, K. (Eds.) Ablex Publishing Corp. Norwood NJ.

Polya, G. (1957). *How to Solve It* (Second Edition), Princeton University Press, Princeton NJ.

Scardamalia, M. and Bereiter C. (1986). "Research on Written Composition," in Wittrock, M. Ed., *Handbook of Research on Teaching: Third Edition*, MacMillan, New York NY.

Scardamalia, M., Bereiter C., and Steinbach (1984). "Teachability of Reflective Practices in Written Composition," *Cognitive Science*, Vol. 8, pp. 173-190.

Schoenfeld A., (1987). "A Brief and Biased History of Problem-Solving," in Curcio, F. (Ed.), *Teaching and Learning: A Problem-Solving Focus*, National Council of Teachers of Mathematics, Reston VA.

Thyberg, L.F. (1987). "The basis for a Process Approach to Programming in Logo," *Proceedings of the East Coast Logo Conference*, April 1987, Meckler Publishing, Westport CT.

Walker, D.F. (1987). "Logo Needs Research: A Response to Papert's Paper," *Educational Researcher*, Vol. 16, No. 5, June/July 1987.

Watt, D. (1988-89 in press) "Assessing Logo Learning in Classrooms," (a series of monthly articles), *The Logo Exchange*, September 1988 - June 1989.

Watt, M. and Watt, D. (1986a). *Teaching With Logo: Building Blocks for Learning*, Addison-Wesley, Menlo Park CA.

Watt, M. and Watt, D. (1986b) "Collaborating With Teachers to Evaluate Logo Learning," *Proceedings, Logo '86*, MIT, Cambridge MA.

Watt, M. and Watt D. (1987b). "Polishing Programs for Publication: A Neglected Step in the Logo Programming Process," *Logo Exchange*, May 1987.

Watt, M. and Watt D. (1987a). "Step by Step," *Proceedings of the East Coast Logo Conference*, April 1987, Meckler Publishing, Westport CT.

Watt, M. and Watt D. (1987c). "Using the Process Approach to Teaching Writing as a model for Teachers of Logo," *Proceedings, Third International Conference on Logo and Mathematics Education (LME3)*, July 1987, Concordia University, Montreal Canada.

Watt, M. and Watt D. (1988). "Teachers as Collaborative Researchers: Professional Development Through Assessing Logo Learning," *Proceedings, National Educational Computing Conference*, June 1988, Dallas, TX.

Watt, M. and Watt D. (1989). "Professional Development of Logo Teachers Through Collborative Classroom Research," Final Report: Exploratory Research on Critical Aspects of Logo Learning (NSF Grant # MDR 865 1600). Education Development Center, Newton MA.

Figure 1a

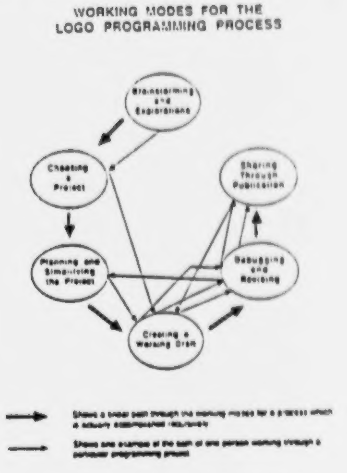

WORKING MODES FOR THE LOGO PROGRAMMING PROCESS
Brainstorming and Explorations
Choosing a Project
Sharing Through Publication
Planning and Simplifying the Project
Debugging and Revising
Creating a Working Draft

Figure 1b

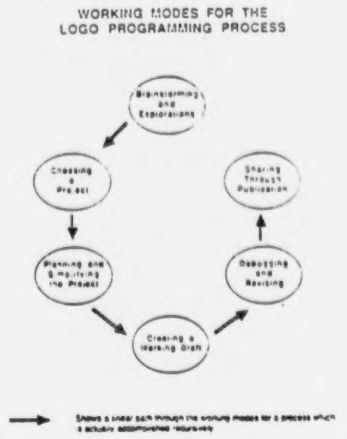
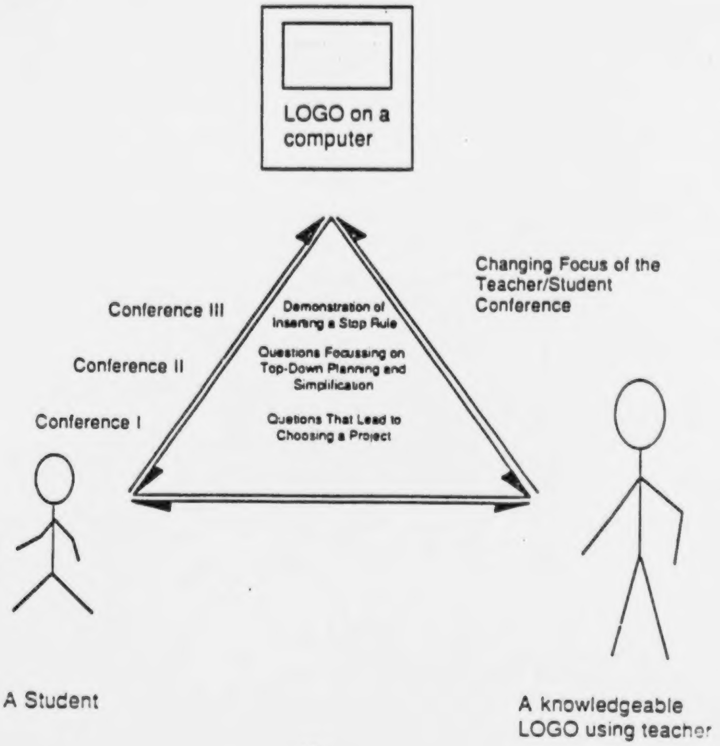

WORKING MODES FOR THE LOGO PROGRAMMING PROCESS
Brainstorming and Explorations
Choosing a Project
Sharing Through Publication
Planning and Simplifying the Project
Debugging and Revising
Creating a Working Draft

Figure 2


LOGO on a computer
Changing Focus of the Teacher/Student Conference
Conference III
Conference II
Conference I
Demonstration of Inserting a Stop Rule
Questions Focussing on Top-Down Planning and Simplification
Quetions That Lead to Choosing a Project
A Student
A knowledgeable LOGO using teacher

MATHEMATICS EDUCATION: LEARNING FROM THE PROCESS APPROACH TO WRITING

Rebecca Corwin and May C. Reinhart
Lesley College
Cambridge, MA

An interesting trend in teaching writing, the writing process approach, has emerged in this decade, dominating the field of the teaching of writing. Unlike trendier approaches to curriculum change, it is solidly supported both by research and by teacher acclaim, and because of that has become very popular in classroom practice. The writing approach was introduced to the national consciousness through both grassroots and top-down change efforts, and has been widely adopted and practiced (Applebee 1986).

Mathematics teaching has been given negative press for nearly twenty years, with little agreement about what would best constitute effective reform of the field. The publication of the first draft of the recommended Standards of Mathematics Education marks the first overall attempt at revision of the math curriculum since the New Math of the early nineteen-sixties (National Council of Teachers of Math 1987). It is our contention that there is much for mathematics educators to learn from the writing process approach as a model of instruction, and that its assumptions and its recommended approaches dovetail well with proposed revisions of the mathematics curriculum.

In this paper, we describe the writing process and show similarities between teaching children to be writers and teaching children to be mathematicians. For the purposes of illustration, we take examples from "Used Numbers", a K-6 statistics and data analysis curriculum currently being developed at Technical Education Research Centers, Inc., with National Science Foundation funds. We believe that this approach is not limited to mathematics, but can serve as a good model for the revision of most curricula in the elementary and middle schools.

Just as the process approach to writing is modelled upon the approaches most commonly used by professional writers, a process approach to mathematics must focus on having children **do** real mathematical work. Speculating about patterns, solving real problems, and using mathematical processes to spin theories and develop hypotheses constitute **real** mathematical work. We believe that the excitement of doing such **real** work is developed by using a process approach to mathematics, just as writing teachers can create magic in their classrooms using that approach to writing.

The Writing Process

The writing process was not artificially created to deal with teaching children how to write. Rather, it provides the time for children to engage in the series of steps **all** writers take from the time they first select an idea to write about, through the composition and rewriting of multiple drafts, until the final version of a piece of writing is complete. Most descriptions of process approaches include four phases. First, there is a **planning phase** (often called pre-writing). This is followed by the **writing phase,** when a draft of ideas is first put down on paper. The third phase is the revision or **rewriting phase** when the writer elaborates, clarifies, restructures and edits the piece. The final phase is called the **publication phase,** when the writer's completed piece is shared with others. These processes are thought to be recursive rather than linear in nature, and sometimes there is a variation in the emphasis and amount of time spent at each phase (Flowers and Hayes 1981).

One main focus of the **planning phase** is on selection of topics. Unlike methods of teaching writing in which students are assigned subjects to write about, the process approach maintains that everyone--including very young children--has experiences and interests from which to select writing content. Some teachers build pre-writing experiences by scheduling whole-group and individual discussion of possible topics for writing, brainstorming topic lists, keeping journals and records. Their topics come from their own concerns and experiences, both internal and external. Self-selection of topics gives students an active role in exploring their own ideas and in establishing ownership and control over them. Graves (1983) notes that students who are never given the opportunity to select their own topics for writing "remain forever on writing welfare", that is, they become totally dependent on teachers to tell them what to write about.

During the **writing phase** the writer begins to form ideas. Unlike writing approaches which direct students to write a story or a descriptive paragraph, teachers using a process approach to writing rarely dictate what form students' writing should take. Neither do they require students to plan or outline ideas before writing. Teachers who take a process approach see the first attempt at writing to be a time when writers uncover new ways of thinking about a topic. Murray (1984) calls this first attempt to write down ideas the "discovery draft".

Although **revision** occurs during the composing process as writers make and change ideas about work and sentence choice, order, and the like, advocates of a process approach describe a new social or communal dimension for the revision phase of writing. In most traditional writing procedures, students turn in their papers to teachers after a first draft attempt. Often the focus of change is on the improvement of surface conventions of written English, stressing spelling, punctuation, and paragraphing. The paper is then returned to the student for correction. When the corrections are made, the student returns the paper to the teacher. This "tennis match" approach continues until the teacher decides that the final draft has been written; this constitutes a very constricted interaction between teacher and student, and one in which the teacher actually takes most of the responsibility for the work of revision. This represents an extremely limited notion of audience!

By contrast, a process approach to revision centers on ways to help the writer reconsider what is being written. This is done by having children share what they have written with their classmates. As writers read aloud or show their work to others, the listeners respond with ideas and suggestions which the writer might use in improving the piece. Sharing writing with interested listeners underscores the purpose of writing as communication with a wider audience, and not just as an activity that goes on between the teacher and the students. Final decisions about changes in a piece of writing are left for the students, not the teacher, to make.

In the **publication phase** of the writing process, the writer prepares a piece of writing for presentation. Because students work on writing every day and keep their work in folders, sometimes they will work on more than one piece at a time, and some pieces are at a different level of development than others. Graves (1983) suggests that writing is very much like a studio craft in which the artist brings certain pieces to a stage of completion, while choosing to leave others in the sketchbook.

For younger students, publication can mean making books for school or classroom libraries, contributing a poem or story to a school newspaper or magazine, writing a letter to a friend or an official, putting together a guide to the community. As students publish their work, other students read it, and are in turn influenced in the same way adult writers are affected by what they read. In this way, students of the writing process are returned again to a pre-writing stage, collecting ideas and experiences for future work. The phases of the writing process then begin all over again.

Mathematics as a Medium of Expression

In order to consider a process approach to mathematics, we must consider whether it resembles writing in certain ways. Writing is, after all, used to communicate. It is a form of language used to present ideas to an audience; another form of linguistic exchange. Can the same be said for mathematics?

Mathematics is indeed a form of communication. It evolved historically as a specialized language used to describe, to contemplate, to measure, to reflect on specific aspects of the world and our human environment--in a variety of realms, including the imaginary and the aesthetic. It is not at all a static field. The processes in which mathematicians engage, the dialogues, discussions and debates, are seldom even described, let alone shown to children.

As a specialized language, mathematics has both syntactic and semantic components (Burton 1988). That is, it has a grammar of conventions and forms which is based on the meaning of its operations and symbols. It is based on, comes from, and relates directly to, regular English. Mathematics has powerful meaning and communicative power. Sadly, the experiences underlying and representing the symbols and conventions of mathematics are often skipped or hurried in the elementary school program. Students typically seem to consider mathematics to be a static set of rules for computations rather than a way of expressing ideas in symbolic form.

> ...[N]o real language is used simply to manipulate its own words. The power of language is not in the words themselves, but in the use that we make of them to communicate with each other. The words of our language support communication because they are symbols, pointing beyond themselves to things we experience in our world. To be real, language has to be about something.
> (Burton 1988, page 4)

It is not enough for us merely to teach the syntax, or surface conventions, of mathematics. When we do, we teach only a static form of language. English teachers are beginning to abandon the idea that teaching grammar is useful until students have the cognitive ability to engage in meta-analysis of the language. The same is true of mathematics. The idea of teaching grammar as a way to improve written expression is similar to a continual drill on math facts--a focus on surface structure which may come at the expense of considering the meaning or intent or utility of the written piece. Using the appropriate symbols and structures of mathematics (formulas, for the most part), students can communicate only a limited realm of ideas if they cannot use the language to express their own theories and ideas; in order to do that the semantic aspects must be fully developed. Students should be able to debate, challenge, question, discuss, and theorize about the quantitative aspects of their worlds. Mathematical statements must be able to be understood in the same way as any sentence occurring in the natural language.

Mathematics, like all languages, develops within both receptive and productive modes. Students may understand it before they can produce it; they may be able to make sense of another person's use of the language before they can use it themselves. Ultimately, however, they must be able to hear, to read, to receive mathematical ideas from others; similarly they must be able to produce and talk informatively about records of their own observations and ideas. These ideas may be simple or complicated; they may range from a simple bar graph made from Unifix cubes to a complex calculus proof.

It is exciting to imagine school math reflecting the more active aspects of mathematics, involving children in the processes of **doing** mathematics rather than **following** its accepted procedures. Teachers can learn to help students use mathematics: they can help in recording, reflecting, and describing real experiences. Unless students see mathematics as a way of developing and discussing their ideas, they will forever ask, "Why do we have to know this?" rather than, "How can we show this idea to someone?"

A Process Approach is a Frame of Mind

Mathematics teaching is at least as rigid as the teaching of writing used to be before the imaginative shift which produced the process approach. Procedures followed in most classrooms are essentially the same as those of the nineteenth century; not yet incorporating what we know about children's development and the results of research on children's learning. Although there are constant efforts at reform in the elementary school curriculum, the underlying model of mathematics teaching is a strong one, with models of viable alternatives much less powerful. This is understandable, since much research points to the fact that teachers teach as they were taught, and most elementary and middle school teachers were not strong mathematics students themselves. In fact, the idea of **doing** mathematics is foreign to most teachers (Russell 1988).

In a typical mathematics lesson, many of the didactic approaches formerly used by writing teachers are followed. The teacher presents a problem. The students then follow the prescribed routines to find the One Correct Answer to the problem, engaging in a form of **"numerical target practice"** which is based only upon Getting the Answer Right. The teacher sometimes coaches the students, telling them whether they are close to the target or far away, giving clues about method and choice of routines. There is typically little or no discussion--there is not much to talk about! Because the entire point of the mathematics exercise is being correct, there is no more dialogue than a textbook or filmstrip would provide. Information travels one way and one way only. The form of a mathematics lesson is too often one of old-fashioned recitation.

Even in a technologically-based classroom many teachers follow the "target practice" method of teaching. Many pieces of educational software, for instance, encourage such target practice and are designed to help students develop correct routines and procedures rather than deepening their understanding. For years we have known that even Logo in and of itself does not create a learning environment which provides support for experimentation and excitement.

It is possible, however, to do things differently--to teach mathematics as an exploratory process rather than as a fixed outcome. Mathematics is a field in which to communicate ideas. The computational routines which most teachers stress can appropriately be left to machines. Students can use technology to enhance active investigation and problem solving, just as the students of writing can use writing to reflect on and to communicate their lives and thoughts, thus developing the ability to express ideas. In both mathematics and writing, curricular advances lie in articulating new goals--that is, a shift from correctness of form to finding forms for communicating ideas. Just as in writing, mathematics is more appropriately a tool for expression than a set of procedures.

A Process Approach to Mathematics?

If mathematics can indeed be taught as a vehicle for communication of ideas, it becomes helpful to explore the phases in the process approach with mathematics in mind. In this section we will do that, looking at each of the four phases in terms of mathematics learning. In order to do this we describe them in terms of work done during the field testing of a data analysis curriculum called "Used Numbers" (currently under development at Technical Education Research Centers, Inc. under a National Science Foundation grant). The example which we will use is drawn from the end of the fifth grade curriculum, where students collect data about injuries sustained during play time in their school, look for patterns in the data, and see whether they wish to make any recommendations which might affect the injury rate. This unit on Play Injuries was the culmination of work for this class.

Phase one: Brainstorming and planning

The students in this classroom appeared to be engaged by the problem. They were assigned to ask someone at home about injuries which they sustained in playing, and came in the next day ready to talk about their injuries. Their assignment was to share information and collect data from each other in order to decide how to proceed in order to collect **enough** data.

They brainstormed ideas and defined their terms. Some of their concerns: What was an injury which was big enough to count? What questions would they ask their peers? Should they try out their questions on themselves and then expand to a larger sample of the school? During this time, students discussed, debated, thought about possible sources of data, ways of recording it, and how they would organize themselves to collect information. Of course, during this step in the process they talked a great deal about Gory Things and Bloody Accidents--something that has high interest for most fifth graders.

After a great deal of discussion the students agreed on the procedures they would use to collect information from their peers. Equipped for that collection, they took their questions to the larger school population.

Phase Two: Composition

Collection of data allowed the students to being to develop their "discovery drafts"--what we call "quick sketch graphs"--the first draft of the information on which they would base their developing theories. During this phase of data analysis, students spent time looking at the data which had been tallied and translating that data to graphs so that they could condense it. They began to sort out some patterns in the tallied data, but the sense of the group was that they needed to condense the information, and they decided to graph the information because they felt it would help them see patterns.

As students set up and developed those graphs, they were working on producing ideas in a form analogous to developing the first draft of a written piece. From a welter of different images, impressions, and words they selected and highlighted those they felt would help them to look at the most salient information.

These graphs became the most helpful artifacts which the students produced. The power of telescoped information meant that the graphs were treated with deep respect. The students knew that these drafts could be revised and redesigned as they looked for relationships and patterns, and they adjusted their hypotheses as they discussed their findings with their classmates.

Phase Three: Revision

Writers are encouraged to share their drafts with audiences in order to determine how the audience perceives their work. Similarly, in the data analysis process, the students present their draft graphs to the group in order to see whether their interpretations of the data seem correct to others, and whether others have questions which need to be followed.

After spending time making and analyzing preliminary representations of their data on injuries, the fifth grade students presented their first draft graphs to the rest of the class. They met as a whole group to look at patterns which groups had discovered in the data. ("There have been an average of 4.2 accidents per year and that means that by 1990 there will be 8.4 more." "There are more injuries on the big yard than on the small yard." "Most accidents were caused by running." "Supervised games don't have as many accidents.") Students showed their graphs to the whole group, made their statements, and were open to hearing from others if there were different interpretations of the information.

This part of data analysis is essential to statisticians as well. Just as an author needs to have an audience in mind for a story or essay, a person who creates graphs and charts with an eye to using them to generate and illustrate hypotheses must try out both information displays and first draft interpretations in order to consider alternative hypotheses.

Like the writing process, this part of the math process can be very recursive. As an example, consider the events in a fourth grade class, where data describing the class were collected, bedtimes for boys and girls were graphed. During this sharing and revision process, students immediately became deeply involved in generating alternative hypotheses to explain why girls' and boys' bedtimes might be different. These ranged from the physiological ("boys need less sleep") to the sociological ("girls complain less and do what they're told"). Spontaneously, they decided to test the "boys need less sleep" theory by collecting information about the time boys and girls had to get up in the morning. They decided that they needed more information about how much time each of the members of the class spent in sleeping. Students showed their skills in questioning, re-collecting, and revising theories in the face of new input. This discussion and debate and the resulting re-investigation of the question would not have happened in the absence of a presentation to an audience of peers.

The critical piece of this phase is, then, the sharing of first graphs and interpretations or theories with the whole class. In this way the graph is treated not as a finished product, but as a quick sketch of the data, and a display for information. After a thoughtful class discussion of their theories, students return to their work a with new ideas about interpreting and displaying data. The graphs are truly first drafts, just as the first version of a story or an essay is a first draft--genuinely, in Murray's terms, a "discovery draft" (1984).

Phase Four: Preparation for publication or display

The publication of the results of math work varies as it does in writing a story or essay. Students in the classroom studying play injuries made recommendations to the principal and assistant principal based on the conclusions they drew from their study. In order to make those recommendations especially powerful they decided to present them in written form, illustrated with graphs, to the principal and the assistant principal. This involved making final presentation graphs of information previously displayed in working graphs, writing out a description of their study, their conclusions, and their recommendations, and a verbal presentation to the school administrators.

Students who learn data analysis skills can apply them to issues which directly affect their lives. They often develop theories about a topic which leads to action in the real world. When students collect information about real phenomena they can develop recommendations on which others may wish to take action. The presentation of their information in a final form helps them to understand and make sense of the presentation techniques which others use in magazines, newspapers, and on television.

Themes common to these process approaches

Process approaches do not emphasize the production of written products, but rather focus on the thinking and problem solving students engage in as they develop ideas through writing or graph-making. How, specifically, do these approaches in writing and math, two such different subject areas, reflect common themes?

Respect for the learner:

A hallmark of the process approach is a deep respect for the learner's prior experience and the learner's theories. A young writer is treated as a fellow craftsperson by the teacher--a young data analyst as well is treated as a colleague in investigation. Emphasizing process calls upon the learner to relate school knowledge and school work to everyday events and the concrete experiences of real life.

The process approach thus gives a great deal of control to the learner, focussing on weaving experiences into new representations. A teacher who focuses on the process of communication is automatically oriented to helping the learner to make sense of her experiences, to vest them with meaning, and to communicate those experiences and her feelings about them to another person or persons.

Doing writing becomes the focus of a process approach. Although the teacher has many technical skills to transmit to the learner. the primary focus is not on the mechanics of expression but on the means. Are the words vivid? Do they convey the experience which the student wants to emphasize? What is she trying to say? Similarly, when a process approach is taken to mathematics, the emphasis becomes the collection of information which will help the learner focus on salient aspects of her experience. What patterns does she see in the data? How does she make sense of the welter of information collected? What real situations do these data represent? Can she use the data to support her reasoning about choices? Are her graphs readable? How does she interpret them?

New teacher role:

We assume that students come to the mathematics classroom with a firm base of prior knowledge and assumptions, most of which are not about the mathematics which is taught in the schools but rather about everyday, sensible things. The chasm between home and school knowledge in mathematics has long contributed to teachers' inability to match their examples with student-constructed theories about the way the world works (Ginsburg 1986). Students have developed their own theories about mathematical operations, procedures, and symbols. Often these map onto school mathematics procedures; just as often they do not. When they do not, the student makes what we term mistakes. Generally, however, students' mistakes are intelligent ways of making sense of what they have learned. Assuming that students' mistakes are sensible creates a different role for the teacher than that of corrector; rather the teacher becomes a thoughtful supporter and evaluator--more like a caring colleague than a punitive authority.

It is the teacher's role to find out how children already think about mathematics and to support them in further developing and refining their use of the mathematical language. Teachers who expect children to talk about what they are doing, to develop language as a specific expression of ideas, and to make "intelligent mistakes" are teachers who can support children's learning of mathematical process. This requires a tolerance for and enjoyment of peer group work--not always easy, since it can be untidy and unpredictable. The teacher's role in treating mathematics as communication is to match students' experience with increasingly symbolic modes of communication. This respects students' prior experience and focuses on the meaning which they bring to the symbols of mathematics.

The teacher who wishes to develop a process approach to mathematics must also help students by conducting discussions about their theories. Without the social environment, students' reasoning is not tested. They must be encouraged and supported in presenting and defending their theories of how to understand the world; mathematics is only one of the tools available. Research leaves us with little question that students who are encouraged to think about their own and others' theories are, in the long run, apt to change their own mental models in the direction of the "school models" of learning (Ginsburg 1986).

The teacher must develop a mathematical environment which engages students' natural interest in patterns, numbers, shapes, objects, and solving problems about them. As children begin to talk about the things in this environment, the teacher then takes the expressive language which emerges and begins to develop mathematical conventions which will ultimately permit a student to be a fully functioning member of the mathematics culture.

A process approach to teaching and learning is based on a deep respect for the intellectual integrity of the learner. Just as teachers of writing respect the writing process as an attempt to communicate ideas, sensitive teachers in all areas of the curriculum are coming to see that the core of all disciplines is communication.

Taking enough time:

Forming and shaping ideas so that they are expressive of the individual's original communicative intent takes a good deal of time. In our test teaching we have watched children struggle to make their charts and graphs tell precisely what they want to communicate, inventing and creating complex notation systems as more and more elaborate information needs to be shown. In most mathematics classrooms, however, we rarely see the student allowed to take time, to communicate an idea. Generally the focus is on overlearning and automaticity so that test scores will be adequate. This currently seems to take place at the expense of meaning, speculation, and magic. Of course automaticity is important, but it is useful **hand in hand** with understanding--not standing in its place. Understanding develops with enough time spent in an appropriately responsive learning environment, with communication and expression of ideas emphasized.

Exploring students' ideas over time is largely unfamiliar to mathematics teachers. In general teachers of elementary school mathematics see their jobs as transmitting socially determined content to their students. They do not, by and large, speculate about mathematics and mathematical content themselves, and they do not usually have models in their lives for doing so. [For a strong alternative mathematics teacher model and his lasting impact on a teacher, see Kohl (1984.)]

Taking time is the hallmark of an education which focuses on the making of meaning. Teachers who rush students find, inevitably, that they are not able to apply what they learn to new situations. Teachers who expect students to write on a particular schedule, to finish their stories in one class period, are often disappointed with the quality of the output from their writing classes. Likewise, if students are not given the time to discover the limits of their theories about number, if they are not encouraged to discuss, debate, and argue about problems, they will not generate a rich array of alternative models of the systems which they must understand concretely.

Even for teachers, the issue of taking time for learning is a serious one. How can a classroom teacher afford to take a week on one data analysis problem when there is a set amount of curriculum to cover? We know that real-life constraints abound, yet we also know that the time must be taken. Just as the production of three good essays and two interesting short stories is better in a writing class than the production of fifty mediocre and boring pieces, serious and sustained student work on two interesting real life problems will call upon more skill than superficial work on fifty trivial problems. The shift is worth it; the exciting movement in mathematics education is toward this process approach. Current movements for curricular reform are predicated on movement toward processes.

The Essential Role of Technology in Process Learning and Teaching

There is little question that technology can play an important supportive role in the success of the process approach to writing (Daiute 1986). The word processor allows students to become more powerful writers, because it allows the mechanics of writing and editing to take the background to the creation of ideas. Many students are able for the first time to see their work published so that it looks "real". The excitement of creation and the joy of being able to create different drafts by moving text simply from one place to another have become almost commonplace in many classrooms. It is important, however, to note that the writing process occurs whether technology is used or not!

Mathematical processes are also supported by appropriate technology. Many of the problems which children find appealing in mathematics and the sciences deal with real data. These data are messy, as real data always are. The computer allows students to experiment with messy data, to explore their results, and to show results to each other so that they can easily reflect upon the data in order to think about real issues (should the cafeteria seats be so close together? how can we make our playground safer? what seems to be typical of fourth graders in our school? how much garbage does our town generate in an average day?).

Technology-using teachers appear to be very aware of the fact that the computer can enhance the malleability of mathematics and science--that is, it allows and encourages students to manipulate data and use number displays to illustrate and support their discussions and debates. Hypotheses can be generated from data, tested, built on, abandoned, or revised. Students can engage fully in the act of doing mathematics rather than the frustration of having to have simplified examples or contrived data sets.

The software which has proven very effective in our data analysis project is, not surprisingly, tool software. The database and appropriate graphing software has freed students and teachers to try different representations of data, to add in their own data, and to speculate about the effect of adding different entries. In one unit a sample of twenty-six cats is provided in two ways--through photographs and in the computer database. This includes statistics about the cats--their weights, body lengths, tail lengths, ages, coat colors, eye colors, pad colors--as well as photographs of them. Students are encouraged to analyze the data: first from the photographs and the written labels on the photos; second from the information provided in the computer database; and third, by adding their own set of data to see whether early generalizations and patterns seem to hold. The fact that the students can add data to the database and keep exploring what's typical of cats means that they can manipulate data which has real, personal meaning for them.

Appropriate graphing software has allowed students to avoid some of the tedious aspects of taking a quick look at a sketch of data display. For the students who have used the graphing programs to explore display possibilities, this tool becomes a revelation, allowing them to bounce back and forth between displays, to change displays relatively quickly, to generate more than one display of the same set of data.

Further, we have just begun to work with spreadsheets with elementary school students. They have tremendous potential for students to use in determining some statistical measurements. We move with caution here, since our work to date tells us that there are real problems for many students in terms of describing data statistically. In our experience, it takes a long time to develop some of these concepts so that they have real meaning to students. Unless students deeply understand the standard ways of displaying data, unless they can invent their own methods and interpret them to others, the "landmarks" in the data which have been agreed upon mathematically will tend to continue to be meaningless to them. When students realize the power of the spreadsheet, they love to use it. Better, in our view, that they understand that power because of their understanding of the concepts and their desire to move beyond other forms of calculation.

The computer has indeed the ability to free students to find and work with data sets which are not the tidy, textbook sets. The calculator likewise frees them from the tyranny of calculations which stretch endlessly between the conception of a problem solving method and its realization. However, we must be certain, as we must with any tool, that it is in service to the processes of learning and teaching the concepts that we use it. Students must know that they can test theories using the computer or the calculator; they must see them as helpful tools before they can use them confidently. Then, and only then, will the power of technology truly free the student from the bondage of arithmetical calculation.

Using the calculator and computer in the mathematics classroom has a very salutary effect on instruction, allowing the teacher to let go of some of her predetermined notions of role and power. Using powerful technology allows children to speculate, develop theories, and test their ideas in real time, because of the number crunching power of computers and calculators. Real problems with messy data can be included in the students' work; they can solve real problems in no more time than laboriously plotting unrealistic, tidy problems supplied by the teacher.

As the nature of the problem sets shifts from the textbook ["What is the mean length of these five rivers in South America?"] to reality ["What is the typical height of a student in our class?"], student interest also shifts onto caring about the results of the investigations. As the problems become more complex [exploring the nature of injuries sustained on the school playground and making recommendations for improving the accident rate], student engagement in those problems increases. Interest in results increases dramatically, and along with that a willingness to talk about those results and to consider alternative views of a situation. Interest in results increases dramatically, and along with that a willingness to talk about those results and to consider alternative views of a situation.

Technology and the use of powerful tools allows students to see and represent data in powerful ways: graphing software, spreadsheets, databases, CADD, and specific microcomputer environments such as the *Geometric Supposer* or *Logo*. The idea of using the computer as an object to think with is peculiarly exciting; it allows the teacher to participate in the "first-time-ness" of the technology and of mathematics. Just as writing teachers can share their students' working time by writing alongside them, mathematics teachers can share their students' efforts by becoming colleagues in problem solving. Mathematics teachers have much to learn about functioning as colleagues, assuming the students' perspective, working alongside the student; and they can learn a great deal about it from their colleagues in the writing programs.

References

Applebee, A.N. Problems in process approaches: Toward a reconceptualization of process instruction. In *The Teaching of Writing*. Petrosky, A.R. and Bartholomae, D. (Eds.) Chicago: NSSE Yearbook, 1986.

Berthoff, A. (1981) *The making of meaning.* Montclair, New Jersey: Boynton/Cook.

Bizell, P. (1986) Composing processes: An overview. In Petrosky, A.R. and Bartholomae, D. (eds.) *The teaching of writing,* Yearbook of the National Society for the Study of Education. Chicago, IL: University of Chicago Press.

Calkins, L.M. (1986) *The art of teaching writing.* Portsmouth, N.H.: Heinemann.

Daiute, C. (1986) *Writing and computers.* Reading, MA: Addison-Wesley Publishing Co.

Flowers, L. and Hayes, J. (December 1981) A cognitive approach to writing. *College Composition and Communication.* *32*: 365-87.

Ginsburg, H. (1986) *Children's arithmetic: How they learn it and how you teach it.* Austin, Texas: Pro-Ed.

Graves, D. (1983) *Writing: Teachers and children at work.* Exeter, N.H.: Heinemann.

Jackson, L.A. (October 1986) Nineteenth-century elementary composition instruction. *Language Arts. 63:* 6.

Kohl, H. (1984) *Growing minds: On becoming a teacher.* New York: Harper and Row.

Murray, D. (1984) *Write to Learn.* New York: Holt, Rinehart and Winston.

National Council of Teachers of Mathematics. (1987) *National Council of Teachers of Mathematics: Standards (Draft).* Reston, Virginia: National Council of Teachers of Mathematics.

Technical Education Research Centers, Inc. (in process) *Used Numbers: Collecting and Analyzing Real Data.* 1696 Massachusetts Avenue, Cambridge, MA 02138. National Science Foundation Grant No. MDR-8651649.

TEACHING INTRODUCTORY PROGRAMMING METHODS USING STRUCTURE EDITING: SOME EMPIRICAL RESULTS*

Dennis R. Goldenson
Carnegie Mellon University
Pittsburgh, PA 15213
412/268-2173
drg@cs.cmu.edu

Abstract

This paper reports the results from the first of a series of user studies currently being conducted at several different secondary schools and universities. In it a group of university undergraduates who used a structure editor based programming environment performed a full letter grade better on their final programs and exams than did students in a class using separate editing and compilation tools. By themselves the results are far from definitive, and are subject to many experimental design caveats. But they do at least provide a very suggestive case study about how such tools can substantially improve the early learning and teaching of computer programming methods.

What Ought We Be Teaching?

The value of teaching computer programming has increasingly come under fire in recent years. Enrollments in programming courses are down at many secondary schools throughout the country. The current focus on computer literacy has led to a move towards teaching facility with commonly used spread sheet and database applications programs, while programming methods are often said to be unnecessary and inappropriate for most students.

In fact the quality of programming education has left a great deal to be desired. Failure and drop-out rates are all too high. There is a shortage of properly prepared teachers. Emphasis in first courses tends to focus by default on tedious details of language syntax and tool invocation. Students get bogged down in questions about where to put semi-colons and how to use obtuse editors and operating systems. The programs that they write tend to be short and graded on the basis of whether they "work" with a limited and poorly defined set of possible inputs. Fundamental insights of abstraction and "algorithmic thinking" from computer science and software engineering tend not to be emphasized by teachers nor appreciated by students, at least partly because the programs they write tend to be so short and lacking in complexity.[1]

Yet some of us continue to think that it is possible to do better. Students can handle more difficult assignments, and they can develop abilities to deal with abstraction and write clearly structured extensible programs that solve non-trivial problems. Certainly such preparation is needed for future software engineers. In some ways it may be even more important for liberal education, for literacy in its most basic sense.

One of the problems faced in introductory courses is that the structured languages like Pascal do seem to be somewhat more difficult to teach to novices.[2] Aside from the shortage of teachers who know the language, there are in fact additional syntactic and semantic complexities to the language. Declarations and definitions must be made explicitly, be in the proper location and match with respect to type and class. Much of this is very desirable from the point of view of structure and clarity. But it does impose additional burdens of detail on the students.

What then are computer science educators to do? Work over the past two decades in the development of integrated programming environments based on structure editors [19,16,3,2] is now resulting in commercially available software [4,5,20,15]. The essential idea behind such software is to remove students' attention from extraneous, low-level details of language syntax and tool invocation, such that they can concentrate more fully on issues of program structure and design. The environments combine the functionality of a text editor with that of a compiler and a debugger, all in a single uniform user interface.

There has been a good deal of discussion in programming systems and software engineering circles about the desirability of structure editing. However there has been very little work indeed that attempts to assess its actual effects on students.[3]

This article summarizes a pilot study conducted during the Fall semester of 1987. Its results are reported following a brief description of the environments that were used in the study. By itself the study can in fact be faulted on various methodological grounds. However the results are distinctive enough, and previous empirical research rare enough, that they are worth reporting now.[4]

---

[1] See [8] for a persuasive argument to use large programs early in computer science education.

[2] Such has taken on the status of a truism for some of late. For example one recent MIS text [11, p. 223] asserts that Pascal "is not easy for beginners to use."

[3] However do see [18,13,7].

[4] Initial indications from other studies ongoing at the time of this writing suggest that the basic results in fact will be replicated, in the presence of fewer methodological reservations.

Structure Editing

Two related structure editing environments currently are under development at Carnegie Mellon University for use on Apple Macintosh computers.[5] An environment for the Karel the Robot teaching language is now available publicly through Kinkos[4,14]. A Pascal environment is currently in use in beta test version at several universities and secondary schools throughout the country, and also is available through Kinkos [5].[6]

Called GENIEs both are fully integrated environments including editing, translation, debugging and run time tools. Like earlier structure editors the GENIEs automatically handle low level syntax and formatting details. Beyond that they handle tool invocation details in a much more graceful manner.[7]

As seen in figure 1 the student is first presented a partially complete program skeleton. The system automatically provides required syntax and takes care of formatting and related housekeeping chores. The mechanics of building a program simply require the student to replace "placeholders" with code and/or additional placeholders. Placeholders for optional structures are hidden in optional placeholders and can be expanded when necessary. Code is guaranteed to be syntactically correct if the student chooses one of the construct menu options provided by the environment based on the context of the selected placeholder.

In addition the environments provide alternative views of the program database meant to provide support for understanding higher level aspects of program structure, design and implementation.[8] For example, by using the outline view shown in figure 2 students are able to focus their attention on the procedural decomposition of their programs, without being inundated by implementation details. When it is time to attend to such details the use of the outline view makes it possible to keep the details in structural context while implementing them in another concurrently visible view. Troublesome implementation tasks also seem to become more manageable. For example properly mapping formal and actual parameters is less error prone since both the call and declaration sites can be visible at the same time.

Some Empirical Evidence

A basic rationale behind structure editing is that students' early attention can and *should* be focused away from low level details of language syntax and tool invocation, and *to* higher levels of procedural and data abstraction. But such a software engineering based rationale remains quite controversial in computer science education circles, if not in psychological and educational theory. It also is noticeably lacking in support based on systematic user studies.[9]

What evidence, then, is there to support the use of structure editor based programming environments? In a previous field experiment of in-service teacher preparation, we found that those who used a Karel environment built for the VAX performed consistently better than those who used standard editing and compilation tools [18]. They were better able to evaluate existing code, design and implement well-modularized and executable programs. And they were better able to successfully modify previously existing code to deal with changed program specifications.[10]

Such results are very encouraging for the future of computer science education. However they also are quite limited in scope. Hence we now have embarked on a series of empirical studies to see if similar findings can be replicated and further clarified elsewhere. This article reports the results of the first of those studies.

A Field Experimental Study

As mentioned earlier the current study was conducted using university undergraduates. They completed the first course[11] for computer science majors at their institution during the Fall semester of 1987. One group of students used both the Karel and Pascal GENIEs on Apple Macintoshes. The other group followed the standard first course curriculum using ULTRIX and VI on networked IBM PCs.

Though the two groups were chosen on the basis of administrative convenience rather than randomly or on the basis of prior programming performance, there is some limited evidence that their backgrounds were reasonably similar prior to the course. Data in table 1 from the self administered survey instruments described more fully below indicate that both groups had similar academic class status and previous exposure to computing.

---

[5] An additional environment for the very high level language Ada is also under the early stages of development. See [6].

[6] Finding the requisite mix of equipment, qualified teachers, and numbers of students to make the statistics interesting has proven to be much more difficult than we had initially anticipated. Please contact the author if you know of or are interested in becoming a test site.

[7] See [13,7] for discussions about how structure editing user interfaces can be clumsy and/or counter intuitive.

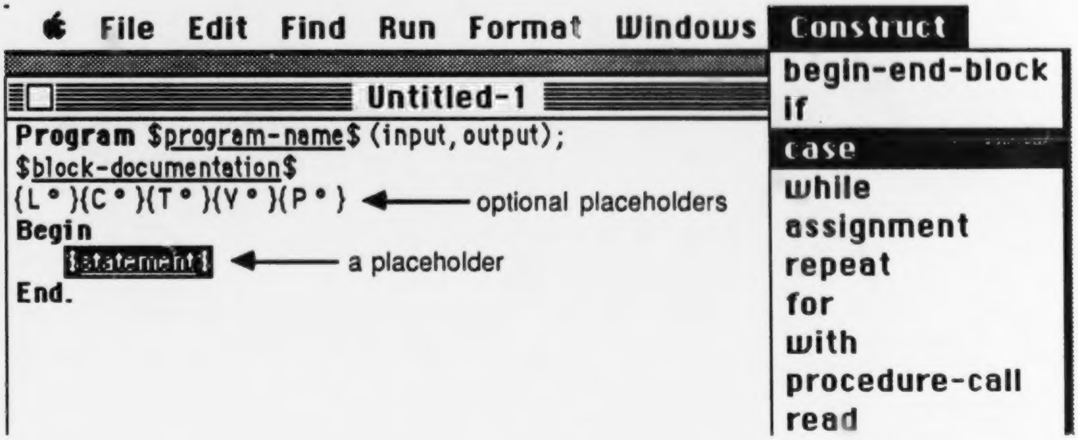
[8] See [2,3,16] for further discussions of alternative program views. In a related vein see [17] for a description of a GENIE graphical design view of the program database.

[9] What evidence there is tends to suggest that developers need to spend considerably more effort at improving the quality of user interface. See [13, 7].

[10] They did do equally poorly on a debugging exercise. But it focused almost entirely on low level language syntax details.

[11] The course generally followed the ACM CS1 guidelines.

Figure 1. Structure Editing Features in a GENIE Pascal Program

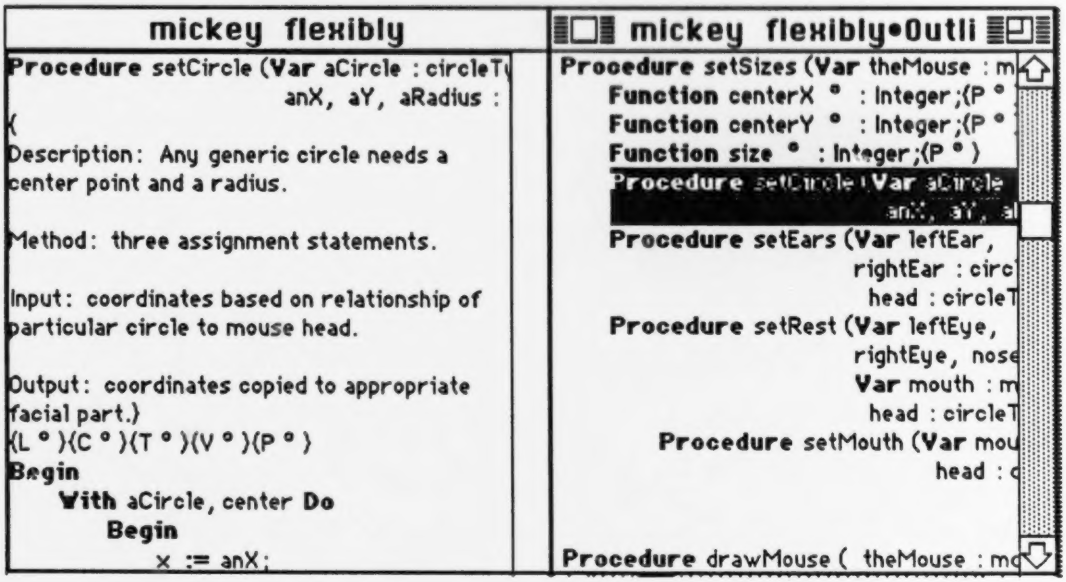

File
Edit
Find
Run
Format
Windows
Construct
Untitled-1
begin-end-block
if
case
while
assignment
repeat
for
with
procedure-call
read

```
Program $program-name$ (input,output);
$block-documentation$
{L ° }{C ° }{T ° }{V ° }{P ° }
Begin
    {statement}
End.
```

optional placeholders

a placeholder

Figure 2. GENIE Pascal Program with Outline and Linear Code Views Open Concurrently


mickey flexibly
mickey flexibly•Outli

```
Procedure setCircle (Var aCircle : circleT
                        anX, aY, aRadius :
{
Description:  Any generic circle needs a
center point and a radius.

Method:  three assignment statements.

Input:  coordinates based on relationship of
particular circle to mouse head.

Output:  coordinates copied to appropriate
facial part.}
{L ° }{C ° }{T ° }{V ° }{P ° }
Begin
   With aCircle, center Do
      Begin
         x := anX;
```

```
Procedure setSizes (Var theMouse : m
   Function centerX  °  : Integer ;{P °
   Function centerY  °  : Integer ;{P °
   Function size  °  : Integer ;{P ° }
   Procedure setCircle (Var aCircle
                           anX, aY, a
   Procedure setEars (Var leftEar,
                        rightEar : circ
                         head : circleT
   Procedure setRest (Var leftEye,
                        rightEye, nose
                        Var mouth : m
                         head : circleT
      Procedure setMouth (Var mou
                           head : c


Procedure drawMouse (  theMouse : mo
```

Table 1. Student Background

	GENIE	ULTRIX/VI
Class Status:		
freshman	30%	25%
sophomore	35	38
junior	25	25
senior	10	11
	100%	100%
(N)	20	8
Family Own Computer?		
%	55	57
(N)	20	7
Number of Prior Computing Courses:		
median	1	1
(N)	20	7
Number of Previous Languages:		
median	1	1
(N)	20	7
Used Computers Outside of School?		
%	32	33
(N)	19	7

However there also were some clear differences between the two groups. They did after all use very different hardware and system software. Though the basic curricula were the same, there were fundamental differences as well. Perhaps most disturbing each class was taught by a different teacher.[12] The first two differences may be less bothersome than the third, and all three are being addressed directly in other ongoing studies elsewhere. For now, while these design inadequacies obviously confound the present analysis, the survey data presented below do lend credence that at least some of the differences on exam and program performance in fact are due to the GENIE environments and the curriculum they support.

Both groups followed the same basic curriculum guidelines. However the GENIE group differed in two fundamental ways. First they completed a three week unit using Karel the Robot before moving on to their work with Pascal. In addition, following work by Bonar [1], they were required to complete written plans prior to submitting their completed program assignments. Like the GENIEs themselves both the Karel language and the formal plans are meant to focus student attention on higher level aspects of procedural abstraction. So one could argue that it was these curricular differences that led to the performance differences between the two groups.

Studies currently ongoing are directly addressing this issue. Regardless, though, one can also argue that the environment itself facilitated the curricular differences. Since the GENIE students needed to spend less time on low level syntax and tool invocation details, they had more time early and throughout the course to devote to other classroom activities meant to develop higher level program design skills.

Performance Results

As seen in table 2 the two groups did differ significantly in their performance on both their final examinations and their final programs. In both instances the group medians were a full letter grade higher for the GENIE group.[13] Moreover students in the GENIE group did considerably better with respect to the quality of their programs' commenting and, especially, their procedural abstraction. Their comments were more complete and likely to state explicitly their program specifications, algorithms and data flow. The GENIE students were much less likely to use repetitive code, and much more likely to decompose their programs in a modular (one conceptual action per subprogram) manner with a clear hierarchical call sequence.

The final exams were completed with paper and pencil at semester's end. The exam included exercises in program tracing, data abstraction, coding according to stated specifications, and extending/modifying program functionality. The programs for both groups were completed on line using ULTRIX and VI.[14] Both the exams and programs were graded blind, without regard to the group differences at the time of grading.

---

[12] Both teachers were rated quite well on balance by their students. The teachers were rated according to the extent to which they were "understandable," "well organized," "available for help," "fair," "interesting," "concerned," and "knowledgeable." Responses were combined in a simple summed index, with possible scores ranging between 7 (most positive) to 28 (most negative). Both teachers received median scores of 10.

[13] Of course such does not argue that the GENIE group's performance was without fault. Disturbingly large numbers from both groups failed to complete either their final exams and/or their final programs. Still, the group performance differences are substantial, and the GENIE group was somewhat more likely to finish their assignments. In the GENIE group 19 of 26 students completed their final exams, while 17 completed their final program assignments. In the ULTRIX group 14 of 23 completed their final examinations, while 12 completed their final program assignments.

Table 2. Performance Results*

	GENIE	ULTRIX/VI
Final Exam[1]**	82%	70%
Final Program[2]***	85%	70%
Documentation**	88%	58%
Procedural Abstraction***	76%	8%

* The final exam and total program grades are expressed as medians for each experimental group. The program component entries are the percentages of each group who did not have points deducted for that category. Statistical significance is assessed using the Mann-Whitney U-test.

** $p < .05$

*** $p < .01$

1 Ns = 24 and 19 respectively

2 Ns = 17 and 12 respectively

One of the common complaints about structure editing is that students somehow will come to rely too heavily on the fact that syntax and formatting support are provided automatically. The concern is that the students will then be lost and inundated by essential details when they no longer have access to these features in their future programming endeavors. Hence it was agreed that the GENIE group in this first study would complete their final programming assignment using the ULTRIX/VI tools that are standard in the rest of their department's undergraduate course offerings.

Our expectation was that learning the necessary syntax and tool invocation details would not be particularly difficult for the GENIE students. Typically students must learn such details at the very beginning of a first course, without the benefit of very much conceptual context. By semester's end, however, the GENIE students already had a conceptual infrastructure with which to organize the details. They had regularly seen syntax and formatting handled properly throughout the semester. Though editing and other aspects of tool invocation differed in terms of their command languages, the functionality is after all quite similar.

The fact that the experimental students were able to perform *better* than the others in what was for them a new environment is telling evidence in support of the rationale for structure editing. The experimental students' early emphasis on structure and design seems to have accounted for their better substantive performance on the exam and programming tasks. Of course it clearly did require some additional effort on their parts to learn the details of required syntax and tool invocation. But they did so successfully.

Survey Results

As indicated earlier data from student surveys can provide some useful insights about the students' satisfaction with and use of their respective programming environments.[15] Data are available from five separate surveys. Both groups of students in the present study completed identical questionnaires at the end of the semester. In addition the GENIE group completed a questionnaire that probed more deeply about their use of the Pascal environment. These two surveys are augmented by data from a short exit poll taken in the context of a laboratory experiment at the same school. In addition, we also conducted two surveys of advanced placement students who used the Pascal and Karel GENIEs in an early admissions program at Carnegie Mellon during the Summer of 1987.

User Satisfaction: Not only did the experimental students perform better on their programs and exams, they were also very likely to express satisfaction with the GENIEs. As seen in table 3 those who used ULTRIX/VI were much more tentative in their opinions. Since the experimental students were required to use both system configurations, we also were able to ask some head-to-head comparison questions. GENIE won hands down. Similarly, the group of students who participated in a short laboratory experiment[16] at the same university also expressed a preference for GENIE, in spite of the fact that they had used rather inflexible versions with their text editing features disabled [10].

---

[14] The experimental group was forced to move to ultrix/vi at semester's end since that system is used for subsequent courses in the university's computer science department.

[15] The survey completion rates are generally high. Of the 26 students enrolled in the GENIE group 13 completed the questionnaire that also was given to the non-GENIE group, while 17 completed the survey specifically on the Pascal environment. All 40 subjects in the lab experiment completed their surveys. Due to a scheduling problem only eight of the 18 enrolled in the non-GENIE group completed their surveys. As indicated above, though, they at least are similar to the GENIE group in their academic class status and prior exposure to computing.

In spite of the small N for the non-GENIE surveys, differences between them and the GENIE group's are often statistically significant according to standard (chi-square) criteria. Of course such criteria are based on individual survey items. The fact that there is so much consistency across a whole series of items is highly unlikely to have occurred by chance. (Remember the multiplication rule.)

[16] The experiment took only one hour's time for each subject.

Table 3. User Satisfaction

All in all how would you characterize this editor/ operating system as an overall "programming environment"?*	GENIE	ULTRIX
very good	29%	0%
good	52	63
poor	19	25
very poor	0	12
	100%	100%
(N)	21	8

Would you use it again if you had a choice?*	GENIE	ULTRIX
definitely not	10%	12%
probably not	10	50
probably would	50	38
definitely would	30	0
	100%	100%
(N)	21	8

Having used both the MacGNOME GENIEs and Ultrix and vi on the VAX, which one do you prefer?**	
definitely the VAX	15%
on balance the VAX	19
on balance the GENIE	33
definitely the GENIE	33
	100%
(N)	21

Which would you use if you had a choice?**	
definitely the VAX	19%
on balance the VAX	14
on balance the GENIE	29
definitely the GENIE	38
	100%
(N)	21

Based on what I saw in this brief experiment, I'd prefer to be able to use an environment like this one in [my regular class].***	
agree strongly	31%
agree	51
disagree	18
disagree strongly	0
	100%
[N]	40

* source: semester end survey to both experimental groups

** source: semester end survey to GENIE group only

*** source: lab experimental exit survey

Using the Environments: Of course, a basic rationale for structure editing is that it provides order and organization early and throughout the first course. Yet structure editing often is faulted for imposing too *much* structure on novice students. Our survey data provide some interesting perspective in this context.

First of all there is some evidence that may help explain why the GENIE group did better on the exam and final program performance measures. Comparisons between the two groups of students indicate that those in the GENIE group found their environment easier to use, and did experience less difficulty with syntax and tool invocation details.

As seen in table 4 the GENIE group reported fewer problems with bothersome details such as matching parentheses, single quotes, comment brackets and begin-end blocks. Perhaps particularly noteworthy they also reported less difficulty with procedure and function declaration syntax and knowing where to insert those declarations in the program's linear order.

In addition students in the GENIE group were more likely to report that they could keep track of the structure of a large program. (See table 5.) Interestingly enough they also reported that it was easier to make minor changes to their code.[17]

---

[17] It is often difficult to make minor changes in program code with structure editors. With GENIE any piece of program structure can be edited textually, making minor changes easy to accomplish. The modified code is then parsed immediately, with any newly introduced errors reported immediately. The ease of minor editing operations may also be due to the nature of the menu driven GENIE/Macintosh interface. It may be a structure editor, but it also appears to be easier to use than an editor based on cryptic commands and key bindings.

Table 4. Reported Frequency of Difficulty with Syntax*

	GENIE	ULTRIX/VI
missing or unmatched parentheses[1]		
frequently	0%	0%
often**	10	38
not very often	12	38
rarely if ever	75	25
	100%	101%
(N)	20	8
missing/unmatched single quotes[1]		
frequently	0%	12%
often**	5	38
not very often	35	38
rarely if ever	60	12
	100%	100%
(N)	20	8
missing/unmatched comment curly brackets		
frequently	0%	0%
often**	5	25
not very often	20	25
rarely if ever	75	50
	100%	100%
(N)	20	8
missing/unmatched begin/end blocks		
frequently	0%	12%
often**	0	12
not very often	40	50
rarely if ever	60	25
	100%	99%
(N)	20	8
procedure/function declaration syntax[1]		
frequently	5%	25%
often**	9	38
not very often	62	12
rarely if ever	24	25
	100%	100%
(N)	16	8
not knowing where to put declarations/definitions[2]		
frequently	0%	25%
often**	5	38
not very often	50	25
rarely if ever	45	12
	100%	100%
(N)	20	8

---

* source: semester end survey to both experimental groups

** "more often than I'd have liked"

[1] $p < .05$

[2] $p < .01$

Data from the survey that probed more deeply on the GENIE groups' use of the environment are also instructive in this context. They suggest that the structure editing features of the environment were in fact quite helpful to the students. Far from succeeding in spite of those tools, the students report relying on them, even when their use was optional.

Table 5. Reported Difficulty in Editing*

	GENIE	ULTRIX/VI
When you're working on a large program how difficult is it to keep track of where you are and what goes where in the editor?[1]		
very easy	5%	0%
pretty easy	75	25
pretty difficult	20	62
very difficult	0	12
	100%	99%
(N)	20	8

How difficult is it to make minor changes in your code?[2]		
very difficult	0%	12%
difficult	10	62
easy	80	25
very easy	10	0
	100%	99%
(N)	15	8

---

* source: semester end survey to both experimental groups

1 $p < .05$

2 $p < .01$

Critics of structure editing often suggest that students would rather just be able to type what they please than to be constrained by the choices imposed by a structure editor. Notice in table 6 that the GENIE students report that they are very likely to rely on the menu that allows them to enter code only for syntactically valid program constructs. When given the choice by the environment few opt to type everything directly. In fact, as seen in table 7, the GENIE students are extremely likely to agree that providing syntactic support is one of the environment's most desirable features. This is so for many of the students even in response to hypothetical questions about features not yet implemented.

Table 6. Text *versus* Structural Editing*

When you have a choice, do you usually use the "construct" menu, or type in your code directly as text?

	Fall	Summer Pascal	Summer Karel
I type directly	23%	24%	10%
I use the menu	45%	16%	43%
I use both, pretty much interchangeably	32%	60%	47%
	100%	100%	100%
(N)	22	25	30

---

* Sources: semester end survey to GENIE group only, and AP/EA Summer Pascal survey.

Table 7. Use of Structure Editing Features*

	Fall	Summer Pascal
Pull-down menus and multiple windows are among the environment's most useful features.	73%	80%
A very important feature of the environment is the fact that it supplies all punctuation and properly spells reserved words, making it impossible to make syntax errors.	86%	60%
It ought to be possible to enter and edit code however you like, even if that means that you can make syntax and other errors that the environment otherwise prevents.	18%	13%
After you have once defined them, it would be very helpful to be able to choose user-defined names (variables, procedures, functions, types and so on) from a menu.	68%	76%
The Karel environment asks you if you want to insert a new template whenever you first type in a new instruction name. It would be very helpful if the Pascal environment did that for variables, functions, procedures and such.	45%	40%
(N)	17	25

---

* Sources: semester end survey to GENIE group only, and AP/EA Summer Pascal survey.

Table entries are percents of respondents agreeing to each item.

The Future

Suggestive as the present results may be there is a clear need for replicating and augmenting the study. To that end additional studies are both currently ongoing and being organized for the future. In particular we are systematically investigating the nature of interaction effects between the environments and the curricula with which they are used. If at all possible we would also like to include more teachers with varied levels of preparation in computer science education.[18] And, of course, those who do not use the GENIEs should still use Macintoshes.

There are a number of possible curriculum interactions that are amenable to study at the current time. As suggested earlier the role of other structure oriented tools such as Karel the Robot and various pre-programming planning devices are especially interesting.[19] So too is the use of graphics examples that facilitate the altering of topical order and emphasis [9].

---

[18] A definitive, summative evaluation is premature at best. For that we would need many more student and teacher data points and much better experimental control. Such would most probably require considerable funding for equipment and teacher preparation, and is premature while the environments are still under development. However we remain sensitive to the difficulties and do intend to address them directly.

What happens to students when they no longer have access to structure editing tools remains an important question. How will the GENIE students perform in subsequent courses? Are there significant trade offs where better abstraction skills are also accompanied by difficulties with syntax errors? Though the present study did speak to these issues we currently are addressing them more fully.

Perhaps most importantly we also are looking more closely at the effects of specific features of the environments. For example our surveys indicate that students tend not to take full advantage of the alternative views and elision made available by GENIE, even though the students do recognize their potential benefits. We are currently investigating ways of making these features more easily and naturally accessible.

Similarly the surveys indicate that students do not take full advantage of the system's run time features. Work on making program tracing more graceful and adding dynamic visualizations of a program's data at run time is currently underway [12]. These too are amenable to experimental examination, as are many other aspects of structure editing. We hope that this research will both inform the development of the environments and serve to make the benefits of structure editing more widely known to the computer science education community.

References

[1] J. Bonar and E. Soloway, "Pre-Programming Knowledge: A Major Source of Misconceptions in Novice Programmers," *Human Computer Interaction*, 1(2), 1985.

[2] R. Chandhok, "A Structure Editor Platform for Novice Programming Environments," *Proceedings of the Asia Pacific Conference on Computer Science Education*, Shanghai, China, October 1988.

[3] R. Chandhok, D. Garlan, D. Goldenson, P. Miller, and M. Tucker, "Programming Environments based on Structure Editing: The GNOME approach," *Proceedings of the 1985 National Computer Conference*, IFIPS Press, July 1985.

[4] R. Chandhok, D. Garlan, P. Miller, J. Pane, and M. Tucker, *Karel GENIE*, Santa Barbara, California: Kinko's Service Corporation Academic Courseware Exchange, 1987.

[5] R. Chandhok, *et. al.*, *Pascal GENIE*, Santa Barbara, California: Kinko's Service Corporation Academic Courseware Exchange, 1988.

[6] R. Chandhok and T. Gill, "Supporting the Use of Ada in Introductory Computer Science," *Proceedings of 1988 SEI Conference on Software Engineering Education*, Fairfax, Virginia, 1988.

[7] J.P. Chin, K.L. Norman, and B. Shneiderman, "Subjective User Evaluation of CF Pascal Programming Tools," under review, 1988.

[8] M. Clancy, "Large Programs in Advanced Placement Computer Science," *The Computing Teacher*, 12(8), 1985.

[9] A.Z. Cole and E.J. Adams, "An Integrated Methodology for Computer Graphics Instruction in an Undergraduate Computer Science Program," *Proceedings of EdCompCon '84: Computers as Knowledge Delivery Systems*, IEEE Computer Society, 1984.

[10] D. Goldenson and M. Lewis, "Fine Tuning Selection Semantics in a Structure Editor Based Programming Environment: Some Experimental Results," *acm/SIGCHI Bulletin*, 20(October 1988).

[11] K. C. Laudon and J. P. Laudon, *Management Information Systems: A Contemporary Perspective*, New York: Macmillan Publishing Company, 1988.

[12] B. Myers, R. Chandhok, and A. Sareen, "Automatic Data Visualization for Novice Programmers," *Proceedings of the 1988 IEEE Workshop on Visual Language*, Pittsburgh, 1988.

[13] L. Neal, "Cognition-Sensitive Design and User Modeling for Syntax-Directed Editors," *Proceedings of the 1986 Conference on Human Factors in Computing Systems and Graphics Interface*, Toronto, 1987.

[14] R. Pattis, *Karel the Robot: A Gentle Introduction to the Art of Programming*, Wiley, 1981.

[15] P. Rajani, *et.al.*, *FirsTime: for Turbo Pascal*, Red Bank, N.J.: Spruce Technology Corporation, 1984.

[16] S. Reiss, "Graphical Program Development with PECAN Program Development Systems," *Proceedings of the Software Engineering Symposium on Practical Software development Environments*, ACM-SIGSOFT/SIGPLAN, April 1984.

[17] J. Roberts, J. Pane, M. Stehlik, and J. Carrasquel, "The Design View: A Design Oriented High Level

---

[19] One possibility, currently being tested, would be to have some students use Karel while others complete an otherwise similar course without Karel. A spin off project currently underway involves looking for possible transfer effects from structured programming methods to expository writing skills. For that study ninth graders are using Karel the Robot in the context of a language arts course.

Visual Programming Environment," *Proceedings of the 1988 IEEE Workshop on Visual Language*, Pittsburgh, 1988.

[18] C. Scheftic and D. Goldenson, "Teaching Programming Method and Problem Solving: The Role of Programming Environments Based on Structure Editors," *Proceedings of the 1986 National Educational Computing Conference*, ICCE, San Diego, 1986.

[19] T. Teitelbaum and T, Reps, "The Cornell Program Synthesizer: A Syntax Directed Programming Environment," *Communications of the ACM* 24(9), 1981.

[20] B. Templeton, J. Gray, D. Rowley, R. Ferguson, *et.al.*, *ALICE: The Personal Pascal*, Toronto: Looking Glass Software Limited, 1974.

OBJECT-ORIENTED PROGRAMMING IN EDUCATION

Mark Guzdial
The University of Michigan
1302D School of Education
Ann Arbor, MI 48109
Mark_Guzdial@UB.CC.UMICH.EDU

Abstract*

Object-oriented programming is a paradigm for programming wherein the programmer defines and manipulates entities that combine the properties of procedures and data. This paper surveys the literature on the use of object-oriented programming in education, provides a brief tutorial on the paradigm, and offers reasons for the use of object-oriented programming in the classroom.

Introduction

The benefits of having students learn to program computers have been debated for several years. The main conflict is whether the skills learned in programming (e.g., planning, procedural thinking) are transferred to other domains. There are researchers who claim that significant transfer takes place (Papert, 1980; Klahr and Carver, 1988), while others say that there is no substantial evidence of this transfer (Pea and Kurland, 1986; Khayrallah and Van Den Meiraker, 1987).

The type of programming language or the programming language metaphor is rarely considered in these debates. Specific programming languages are often mentioned (e.g., Logo, Basic, and Pascal) but not general types of languages as in functional or imperative programming languages. The question of the benefits of an individual paradigms is the subject of this paper.

This paper presents a survey of literature on a particular programming paradigm, object-oriented programming, and its use in education. Section III is a brief tutorial on this paradigm Finally, section IV has discussion on the possible benefits of object-oriented programming in education.

Previous Work

There are many object-oriented programming languages in use today. The most popular of these is Smalltalk, developed by researchers at the Xerox Palo Alto Research Center (PARC). Others include KEE, Loops, Flavors, and Object Logo (Stefik and Bobrow, 1986). In addition, object-oriented extensions have been added to many procedural languages (e.g., C and Lisp).

The first use of object-oriented programming in a classroom was in a study implemented by researchers at the Xerox PARC at a middle school in Palo Alto, California (Goldberg and Kay, 1977, and Goldberg, 1978). More than 200 students used their language, Smalltalk, in these studies. The researchers' suggest that these students clearly understood and used object-oriented programming to produce complex applications.

Notably, Goldberg (1978) reports, "Each of the students was adept at making changes...and adding new capabilities to existing class definitions. They had good control over the class properties, knowing where changes had to be made. The students were weakest on iterative methods used for scheduling objects, especially those in which the iteration counter is used as a variable in the computation." Classes are the classification mechanisms in Smalltalk. This observation suggests that classification is easy for students in Smalltalk, but Smalltalk users still face the application problems (e.g., choosing a looping structure or strategy) with which other novice programmers have difficulties (Perkins et al., 1986. and Soloway et al., 1982.)

Carleton University has begun teaching object-oriented programming in four of their Computer Science classes (Pugh et al., 1987): data structures, programming languages, software engineering, and compiler construction. The classes have not been a total success. Pugh et al. report that students in the compiler construction class prefer learning compiler construction for traditional languages than for object-oriented languages. The authors did observe a better understanding and use of data abstraction than was seen in the previous incarnations of the courses, but no supporting data have been collected.

Kempf and Stelzner (1987) have written about teaching the KEE object-oriented programming system, but their paper discusses the teaching methods used rather than the effects on their students of an object-oriented language.

Object-oriented Logo implementations offer use of the paradigm on available microcomputers. An object-oriented Logo is now available for Apple Macintosh computers, called Object Logo (Drescher, 1985) which is produced by Coral Software. No formal reports of observations or systematic research of Object Logo users have yet been published. At least two sets of object-oriented extensions have been created for the Logo programming language on microcomputers. OOLOGO (Guzdial, 1985) based on Smalltalk and Poo Logo (Muir,

---

* The author gratefully acknowledges the support of a University of Michigan School of Education Merit Award during this work. Thanks also to Robert Kozma and Barbara Ericson for reviewing early drafts of this paper.

1986) based on Object Logo ran on IBM PC's and Apple II's. Neither have been studied in the classroom.

There are other object-oriented programming languages now available inexpensively on microcomputers. For example, Digitalk markets a version of Smalltalk for the IBM PC and the Apple Macintosh called Smalltalk/V. Apple's new HyperCard application tool contains a object-oriented programming language called HyperTalk. There aren't yet any published studies on the use of these languages by students.

Basic Concepts of Object-Oriented Programming

This section of the paper describes object-oriented programming with a slant toward Smalltalk-style object-oriented programming. Other languages, such as Object Logo, have a slightly different definition or implementation of object-oriented programming. For example, languages like Object Logo do not have classes (Lieberman, 1986.) The benefits of object-oriented programming in education do not hinge on these differences.

Objects and Classes. Object-oriented programming is programming by defining and manipulating objects. Objects are computational units that have structure (properties, or instance variables) and behavior, that is, data and procedures. Objects that share the same structure and behavior are said to belong to the same class. Members of the same class are called instances of the class. A class definition (specifying the structure and behavior the class' instances) can be considered to be a template for instances.

Stefic and Bobrow (1986) provide a good definition of objects with respect to object-oriented programming. "The term object-oriented programming has been used to mean different things, but one thing these languages have in common is objects. Objects are entities that combine the properties of procedures and data since they perform computations and save local state. Uniform use of objects contrasts with the use of separate procedures and data in conventional programming."

Real World Analogy. An analogy to the real world may be useful. Consider the class of Dogs. Rover and Spot are instances of the class of Dogs. Rover and Spot share properties that all dogs have such as fur and a tail. Rover may have fur that has a value of brown, and Spot's fur may be black, but both have the property fur. Both Rover and Spot understand how to perform certain behaviors that all dogs know how to do, such as digging a hole and wagging one's tail. Rover and Spot can be modeled in an object-oriented programming language as objects that are instances of the class Dogs.

Messages and methods. In most object-oriented systems, object behavior is triggered through use of a message, perhaps with arguments. When an object receives a message, it determines the appropriate response to that message and executes the procedure, called a method, corresponding to that message. This method may use or manipulate the properties of the object receiving the message. It may also send other messages. The triggering message and the method to be executed in response are part of the definition of a class.

About message sending, Stefik and Bobrow (1986) say, "Message sending supports an important principle in programming: data abstraction. The principle is that calling programs should not make assumptions about the implementation and internal representation of the data types that they use." If the only interface to the object is through messages, the object becomes a black box to the programmer once the object is shown to be working correctly. One doesn't have to worry about the internals of a black box, so those internals can be forgotten and thus the programming task is made easier.

Inheritance. Class definitions are arranged in a hierarchy or graph in many object-oriented systems. A class that is a child of another class (its parent) is said to have the same properties and behavior as the parent, except for where it is explicitly defined to be different. Passing structure and behavior in this way is called inheritance.

The main purpose of inheritance is to ease class definition. By creating the class Dog as the child of the class Mammals-with-Tails, for example, the creator of the class Dog already has many properties and behaviors defined. The programmer need only specify the ways that a Dog is different from the generic Mammal-with-Tail and add those properties and behaviors that are unique to Dogs.

Example. The following is a portion of a class definition, based on one appearing in Goldberg & Kay (1976). The notation is a pseudo-code drawn from no particular object-oriented programming language.

This class implements a graphical Box with no parent classes. A new Box (created by a statement like joe := ask Box [new]) will understand all of the messages known to Boxes. Each new instance will have its own properties (in this case, each will have an x and y position and a size).

Note the use of the variable Self in the methods implementing the messages undraw and grow. Self has the value of whatever object received the message. So, if Joe were an instance of the class Box, and Joe were asked to grow by 10 units (e.g., Joe := ask Box [new]; ask Joe [grow (10)]), then Self will be bound to Joe. By sending messages recursively to Self, a class can even hide information from itself. For example, the method for implementing the message draw can be changed in the definition of Box without changing either undraw or grow.

```
class:          Box                         {Create an object class (template) for a Box}

properties:           x, y,                 {X and Y position of box}
                      size;                 {Size of box}

message:        draw                        {Draw the box}
method:

        {Move a global Logo-like Turtle to the box' position
         then draw the box with the turtle's message 'square'}

        ask Turtle [setposition (x y);  square (size)]

message:        undraw                      {Erase the box}
method:

        {Redraw the box with an eraser}

        ask Turtle [setpen (Eraser)]
        ask Self [draw]
        ask Turtle [setpen (Black)]

message:        grow (newsize)    {Adjust the box' size by the 'newsize' argument}
method:

        {Change the size of the box, then redraw it}

        ask Self [undraw]
        size := size + newsize
        ask Self [draw]
```

Consider implementing the class Diamond as a child of the class Box. A Diamond is a titled Box. Diamonds would inherit all the messages and properties of Box. Diamond would require a new method for the message draw (to turn the Diamond-drawing turtle before drawing the square), but undraw and grow could be used without change since they merely refer to the message draw for object-specific drawing.

The Benefits of Object-Oriented Programming in Education

Object-oriented programming is interesting to educators for at least three reasons: it supports developing computer-based simulations as a learning activity; it may reduce common errors that novice programmers make when classifying components of a program; and it encourages programming in a style that expert programmers have been shown to use empirically. The first reason does not assume that computer programming is a good domain for teaching problem-solving – it provides an argument for teaching programming that assumes that learning programming provides a skill (i.e., programming itself) that is useful in other domains. The second and third reasons assume that there is a need to teach programming, and then suggest object-oriented programming may have benefits over more traditional educational programming languages (e.g., Basic, Pascal, or Logo).

Object-oriented programming for simulation. Shafto (1986) has proposed that programming should be taught to all science and mathematics students as a tool for creating simulations. Kay (1972) has also argued for programming in education as a simulation-creating activity in the name of constructivism. Citing Bruner and a quote by Pavese ("To know the world one must construct it"), Kay claims that the process of creating simulations teaches the students to understand that real-world thing that the students are simulating. By creating simulations, the child is channeling his or her innate desire to affect and change some portion of the world into a creative and productive activity.

Object-oriented programming is a style of programming geared toward simulations. Simply put by Pugh et al. (1987), "The fundamental philosophy of object-oriented programming is that all programming can be viewed as simulation." Object-oriented programming was first used in the simulation language, Simula (Dahl & Nygaard, 1966). Several of the original Smalltalk activities suggested to students were computer-based simulations (Goldberg & Kay, 1977). Use of objects allows simulation data structures and procedures to "be altered...with no impact on a calling program" (Kheir, 1988). Thus, the traditionally complex task of writing simulations is made easy by enforcing information hiding and by providing facilities to create abstract data types.

Object-oriented programming to reduce errors. In a study at the Educational Technology Center (Perkins et al., 1986), novice Basic programmers were studied to determine the kinds of errors that they encountered and when they occurred. The errors were categorized as occurring during one of four levels of programming: decomposition, need formulation, command finding, and application. Decomposition referred to realizing the need for a program component, such as a calculation or getting input. Need formulation referred to formulating what the component should do, as in determining that the input must consist of integers read from the keyboard. Command finding referred to retrieving the appropriate command, and application referred to implementing it with the correct syntax. The percentage of students' difficulties occurring at each level were 10% during decomposition, 25% during need formulation, 18% during command finding, and 47% during application. Though nearly half of the errors occurred at the application level (indicating that this is the area requiring the greatest improvement), a significant portion of the errors, 35%, appeared during classification activities. These activities were where the programmer classified the components of the program and defined what these classifications and components meant.

In the PARC study on the use of object-oriented languages in the classroom (Goldberg, 1978), students were observed to be especially skilled at classification and manipulation of classifications. The researchers attributed this ability to the classification features of the language. The implication is that students who use a language whose use requires the traversal and creation of class hierarchies become more skilled at classification. Data is not available to determine whether this observation implies that the students in the PARC study had fewer classification errors than in the ETC study.

Since classification is one of the characteristics of object-oriented languages, object-oriented programming may aid the novice programmer during the decomposition and need formulation level to avoid errors. Object-oriented programming languages may provide a framework or guide for classification that will aid the student in decomposing the program.

Object-oriented programming as an expert's technique. In a study of experts and novices, Adelson and Soloway (1985) reported that novices plan programs in terms of pseudocode while experts create models and simulate the models' activity. Experts think about their programming problems in terms of components whose activity can be and is simulated in the mind of the programmer. Novices, on the other hand, were tied to the programming language and tended to think about the program in a fashion closely related to the lines of program code that they would eventually produce.

As has been mentioned, object-oriented programming is based on simulation. Providing students with tools designed for simulation may encourage them to think about programming as simulation.

There is little evidence that directly supports giving learners tools based on experts' thinking processes to encourage expert-like thinking, though the idea is intuitively appealing. At least some researchers believe that "media with certain attributes will result in more learning of certain tasks by certain learners, if the use of these attributes short-circuits or models certain cognitive processes or representations that are salient to the task and which the learners can not or do not provide for themselves" (Kozma, 1988). An example of this is reported by Salomon and Cohen (1977) where students' ability to pick out visual cues was enhanced by film presentation techniques. Perhaps object-oriented programming may perform the instructional function of a condition to foster acquisition (Glaser, 1976), leading students to this level of expert programming.

Concluding Remarks

Object-oriented programming is a paradigm that has received little study in education. As has been described here, object-oriented programming languages have valuable assets to offer educators. I suggest that they deserve consideration and study.

Studying these languages will be difficult. For example, there are several different types of object-oriented programming languages (Lieberman, 1986), and few run on computers readily available in the classroom. These factors make it difficult to study the effects of the paradigm as opposed to its specific implementations.

An important question is how object-oriented programming compares with more traditional imperative programming languages (e.g., Basic and Pascal). Generally it is difficult to compare one language against another at a novice level, let alone one metaphor for programming against another. Since there are enormous problems in learning and teaching program (e.g., Soloway et al., 1982; Perkins et al., 1986), guaranteeing that the instruction is complete and effective so that any observed effects are due to the language rather than the instruction will be a challenge. Equivalencing instruction used to teach very different languages so as to compare the effects of the languages and their attributes on students is an even greater problem.

Despite the difficulties inherent in studying programming paradigms and object-oriented programming in particular, object-oriented programming is interesting and should be considered as a programming language metaphor for education.

References

Adelson, B. and Soloway, E. 1985. "The Role of Domain Experience in Software Design." *IEEE Transactions on Software Engineering*. SE-11(11).

Dahl, O.J. and Nygaard, K. 1966. "SIMULA – an ALGOL-Based Simulation Language." *Communications of the ACM*. Vol. 9(9). 671-678.

Drescher, G.L. 1987. "Object-Oriented Logo." In *Artificial Intelligence and Education*. Edited by R.W. Lawler and Y. Masoud. Ablex Publishing: Norwood, New Jersey.

Glaser, R. 1976. "Components of a Psychology of Instruction: Toward a Science of Design." *Review of Educational Research*. Vol. 46(1). 1-24.

Goldberg, A. and Kay, A. (Eds.) 1976. *Smalltalk-72 Instruction Manual*. Xerox Palo Alto Research Center Technical Report SSL-76-6.

Goldberg, A. and Kay, A. 1977. "Methods for Teaching the Programming Language Smalltalk." Xerox Palo Alto Research Center Technical Report SSL 77-2.

Goldberg, A. 1978. "Smalltalk in the Classroom." Xerox Palo Alto Research Center Technical Report.

Guzdial, M. 1985. "Object-Oriented Programming in Logo." Second International Logo Conference: Logo 85. MIT, Cambridge, Mass. 115-116.

Kay, A.C. 1972. "A Personal Computer for Children of All Ages." Draft. Xerox Palo Alto Research Center.

Kempf, R. and Stelzner, M. "Teaching Object-Oriented Programming with the KEE System." *OOPSLA'87 (Object-Oriented Programming, Systems, Languages, and Architectures 1987 Conference) Proceedings*. October 4-8, 1987. Orlando, Florida. 11-25.

Khayrallah, M.A. and Van Den Meiraker, M. 1987. "Logo Programming and the Acquisition of Cognitive Skills." *Journal of Computer-Based Instruction*. Vol. 14(4). 133-137.

Kheir, N.A. ed. 1988. *Systems Modeling and Computer Simulation*. Marcell Dekker: New York, New York.

Klahr, D. and Carver, S.M. 1988. "Cognitive Objectives in a LOGO Debugging Curriculum: Instruction, Learning, and Transfer." *Cognitive Psychology*. Vol. 20. 362-404.

Kozma, R.B. 1988. "Media and Learning: A Paradigm Shift." Draft. The University of Michigan.

Lieberman, H. 1986. "Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems." *OOPSLA '86 Proceedings*. September 29-October 2, 1986. Portland, Oregon. 214-223.

Muir, R. 1986. "NLXual Challenges: Objective Orientation." *The National Logo Exchange*. Vol. 4(9). 15-16.

Papert, S. 1980. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books: New York.

Pea, R.D. and Kurland, D.M. 1986. "On the Cognitive Effects of Learning Computer Programming." In *Mirrors of Minds*. Edited by R.D. Pea and K. Sheingold. Ablex Publishing: Norwood, New Jersey.

Perkins, D.N. Martin, Fay. and Farady, Michael. 1986. "Loci of Difficulty in Learning to Program." Educational Technology Center Technical Report. June.

Pugh, J.R. LaLonde, W.R. and Thomas, D.A. 1987. "Introducing Object-Oriented Programming into the Computer Science Curriculum." *SIGCSE Bulletin*. Vol. 19. 98-102.

Salomon, G. and Cohen, A.A. 1977. "Television Formats, Mastery of Mental Skills, and the Acquisition of Knowledge." *Journal of Educational Psychology*. Vol. 69(5). 612-619.

Shafto, S.A.S. 1986. "The Use of Logo in an Introductory Computer Science Course." *SIGCSE Bulletin*.Vol. 18(4). 196-302.

Soloway, E. Ehrlich, E. Bonar, J. and Greenspan, J. 1982. "What do Novices Know about Programming?" In *Directions in Human-Computer Interaction*. Badre, A. and Schneiderman, B., eds. Ablex Publishing: Norwood, New Jersey.

Stefik, M. and Bobrow, D. 1986. "Object-Oriented Programming: Themes and Variations." *AI Magazine*. Vol. 6(4). 40-62.

MATHEMATICS AND PROBLEM SOLVING WITH LOGO

Bill Craig
Carolyn Knox-Quinn

ABSTRACT: Using Logo to Enrich the Elementary Mathematics Curriculum

Bill Craig

Over the past several years, I have become interested in the use of Logo to extend and enrich the study of topics commonly found in an elementary school mathematics curriculum, especially in grades 2-5. This project consists of sets of procedures, written by me or adapted from others' work, which have been used to supplement the study of topics in geometry, probability and statistics, ratio and proportions, and problem solving. Examples include a Logo geoboard, procedures which simulate the Virginia Lottery games, and procedures which provide experiences in deductive and inductive reasoning. The use of Logo was employed only when it could improve students' ability to make mathematical investigations. A handout containing procedure listings and suggestions for classroom applications will be available.

ABSTRACT: Building Expert Systems in the Classroom: A Problem Solving Project

Carolyn Knox-Quinn, International Council for Computers in Education (ICCE), 1787 Agate Street, University of Oregon, Eugene, OR 97403.

Increasingly teachers want their students to engage in higher order thinking in their classrooms. Computer using educators have been particularly interested in developing electronic problem-solving microworlds within which students can test and build representations of some reality. Construction kits and programming languages like Logo have been tremendously popular with such teachers, but have had a narrow range of effectiveness. A computer program, which provides the student with the metacognitive experience of model construction, using their own English phrases, and applicable across a wide variety of content areas, is needed.

Expert system building tools are being used by college students to facilitate problem solving and higher order thinking in curricular areas. Expert system building shells, available now for Macintosh computers, require no special programming knowledge and can be used to create expert systems across a wide variety of subject areas.

In this project 25 junior high school students learned to use expert system shells on the Macintosh with less than an hour of program instruction. They created expert systems, modelling their own expertise, in the subject of choice.

VIRTUAL REALITIES: FROM THE CONCRETE TO THE BARELY IMAGINABLE

Stephen Marcus
Associate Director, South Coast Writing Project
Graduate School of Education
University of California, Santa Barbara
Santa Barbara, CA 93106

ABSTRACT

A virtual reality is an interactive simulation, wheher of a real or an unreal world. For example, text simulates thought. A word processor simulates paper. A novel simulates experience.

This presentation will illustrate and demonstrate some advanced technology-generated virtual realities, along with the means used to enter them. These will include the Virtual Environment Workstation, the DateGlove, SemNet, A Very Nervous System, and the Rubik Cube.

Each of these applications provides an intriguing look into the future of educational computing—not to mention of arcade games. The "look and feel" of the interface, to say nothing of the innerface, will be transformed from the current surface-and-depth of the screen to a far more indepth immersion into the data/word being generated by the program. New cognitive, emotional, and kinesthetic abilities will be required to learn and apply "computer literacy skills". Keyboarding issues will be the least of our worries.

This session will introduce some of the perspectives, issues, and technologies involved in such developments. One cross-curricular application will be demonstrated that is immediately usable in a variety of curriculum areas.

INTEGRATING COMPUTERS INTO THE STAR SCHOOLS PROJECT

Thomas H. Saterfiel, Chair
Mississippi Department of Education
Jackson, MS

ABSTRACT

The entire educational community is seeking ways technology can be used to improve student achievement. The STAR school program represents the most ambitious effort to test the impact of technology on the educational process. The federal government funded four pilot projects to demonstrate how current technology in the areas of sattelite transmission and telecommunications could be confirmed with computer technology to deliver instructional programs to rural secondary schools where such course work is not currently available. The general concept is called "distance learning". These programs are designed to test the technology in a major implementation across the nation. In general, these programs allow a course to be taught to several hundred students in many different sites across the country with students tied to the teacher through telecommunications and computers. Computers are also used to conduct much of the testing. This system can also be used to deleiver staff development programs to teachers and administrators.

A representative for each of the four STAR school projects will discuss the following points concerning their project:

(A) Goals of the program;
(B) Implementation sites;
(C) Technologies used in the delivery of the programs; and
(D) Future potentials of the technology

Following all the presentations, there will be an opportunity for the audience to ask questions about the projects.

PANELISTS

Pamela S. Pease
Ti-In United STAR Network
San Antonio, TX

Kenneth H. McKinley
Oklahoma State University
Stillwater, OK

Gail Arnal
Satellite Educational Resources Consortium (SERC)
Oxon Hill, MD

Cecelia Link
Technical Education Research Center
Cambridge, MA

SPONSOR: AERA

WHAT CAN YOU DO WITH A MINI-LAB?

Robertta H. Barba
Pennsylvania State University

Is there life after computer literacy? What do you do when you have trained your staff and students to be computer users? What curriculum do you use with a computer literate student body? What becomes of the hardware when you no longer need a large computer lab to teach computer literacy skills?

For 6 years the Albuquerque Public School District has had a computer literacy program. Teachers by the thousands enrolled in Mod I (computer awareness) and Mod II (computer literacy). In turn, thens of thousands of students became computer users. The program was centered around the large computer lab placed in every building and a district developed computer curriculum. Every school in the district was equipped with at least one room designated as the computer lab. Fifteen computers were installed in the lab and students rotated into the lab for computer instruction. Soon a generation of computer literate students emerged. What do you do after computer literacy? What do you do when all the students can use a word processing package, a data base, a spreadsheet, can graph using the computer and can program in a simple computer language?

What happens to the large computer lab? What should we teach in the computer lab? Should we offer more advanced computer classes to all students? Staff and Students had already completed the foundationary courses. Teachers felt that computers belonged in the existing academic programs, computers were instructional tools. Computers should somehow be integrated into the existing curricula.

What do you do with the hardware? Many teachers wanted computer labs disassembled and computers shipped into their classrooms. The easy and obivious solution of what to do with the computers was not necessarily the best solution to the problem. Computer labs held about 15 computers each. The result of moving computers into the classrooms would be that not every teacher in a building would get a computer. Would one computer in a classroom be sufficient to supply the computing needs of a classroom? Would one computer in a classroom become the teachers private word processor and electronic grade book, rather than an instructional tool for students?

A second solution was to allow the computer labs to remain intact and allow teachers to sign up for the lab on some equitable basis. This solution was highly populate with teachers who wanted to use computer assisted instruction to enhance their programs. But, there were some problems. What about the science teachers? Science teachers wanted to use computers as laboratory tools, they wanted to come into the computer lab with their racks of test tubes, burners, and foamy, fizzy laboratory activities. Science teachers displayed very little machine apprehension. They wanted to take the tops off the computers and to connect interfacing devices to them. Computer teachers who for years enforced rules such as "No Food" and "No Drinks" in the computer lab were horrified. How could anyone live with the possible mess, and what about possible damage to the computers from spilled liquids. How would the fire marshall react to a computer lab being used as a science laboratory? The list of problems seemed endless. There didn't appear to be a quick or easy solution to the problem.

The intermediate solution of how to handle future computerizations in our schools rests in several pilot projects to determine the most feasible course of action. One of the pilot projects involved the use of the computer mini-lab. A mini-lab would consist of three or four computers mounted on carts. The labs would be placed in the media centers of the schools. Sign-out procedures for the mini-labs would be the same as for exisiting audio-visual materials. The pilot project was established to so all teachers in the building would have access to the mini-lab and would have an equal opportunity to use the technology . If a school has already purchased computers, the cost of carts to make those computers mobile is minimal.

What are the advantages and disadvantges of using one computer configuration over another? (Fig.1)

It was the purpose of this study to find out how teachers would use a computer mini-lab if it were available as an instructional tool. A computer mini-lab consisting of four computers on carts was placed in a school that already had two large computer labs (each consisting of 15 microcomputers), and 10 classroom with one computer in each classroom. In this school there were 55 regular classroom teachers divided into the four traditional academic disciplines: mathematics, science, social studies, and language arts. There were also art, music, physical education, shop, and home economics teachers in the same building. All of the teachers were observed on a daily basis and their computer using patterns were observed.

When computers become available to teachers in the form of a computer mini-lab, how are they used? The following is a graph of the usage of the computer mini-lab by academic departments at one middle school during a school year (based on 950 hours of observation. Fig. 2.)

Fig. 1 Computer Configurations:
Advantages and Disadvantages

Large Computer Lab.
- Pros:
  - Provides instruction for large numbers of students
  - Provides central storage and security
  - Highly efficient model for mass computerization of students and staff
  - Software sharing and cost efficiency
- Cons:
  - No flexibility for scheduling classes
  - Doesn't meet fire code
  - Conflicts between lab custodians and users

One Computer in a Classroom
- Pros:
  - A computer is always available when the teacher needs it.
  - Provides for daily integration of the computer into the curriculum
  - Computer available for demonstrations and expository instruction
- Cons:
  - Little time for individual instruction
  - Every teacher needs software
  - Computer becomes the teacher's personal tool

Mini-Lab
- Pros:
  - Highly flexible and available
  - Meets fire code
  - Software sharing
  - Cost effective
  - Integrate computer into the regular instructional program
- Cons:
  - Sign - up still required
  - Maximum number of students on mini-lab not yet established
  - Requires flexible classroom management

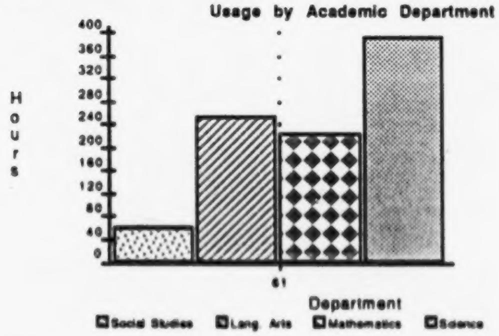


Figure 2.

The science and mathematics teachers were the largest users of the mini-lab, followed by the language arts and social studies teachers. The next question addressed, was what the usage was for the computer mini-lab. Fig. 3 is a presentation of the usage of the mini-lab on a school wide basis:

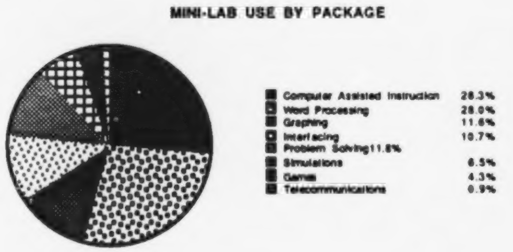


Fig. 3.

Word processing and computer assisted instruction were the most common applications of the computer mini-lab. Some activities such as telecommunications were carried on only by teachers in one academic department. The computer mini-lab proved to be a highly versatile tool. Uses for the mini-lab ranged from the writing of 11-sentence paragraphs in language arts classes, to use as a photospectometer in the science classes and as a graphing tutor in the mathematics classes.

What about the other computers in the school building? How were those computers used? During the observational period the large computer labs were used between 42 and 71% of the instructional day. The one computer in a classroom models were used between 14 and 86% of the instructional day. The computer mini-lab experienced almost 100% usage over the school year.

When the computer mini-lab is used in a school, it becomes a highly versatile tool. The mini-lab is widely used by teachers and students and becomes integrated into the exisiting curriculum. The mini-lab usage, like other computer usage is limited by teacher training, software availability, and points of integration (times when the computer can be easily used in the existing curriculum).

References

Amarel, M. (1983). "Classrooms and Computers as Instructional Settings". *Theory Into Practice*, 22(4), 260-266.

Norton, P. , and Resta, P. (1986) "Investigating the Impact of Computer Instruction on Elementary Students Reading". *Journal of Educational Technology*, 26(3), 35-41.

Schubert, J. (1985). Ideas for Equitable Computer Learning, Palo Alto, CA: American Institutes for Research.

Valdez, D., Einhorn, C., and Resta, P. (1987). Computer Configurations. paper presented at the NMCCUE Conference, Albuquerque, NM.

EDUCATIONAL SPECIFICATIONS FOR A COMPUTER RESOURCE CENTER

Judy S. MacDonald
Instructional Technologies Specialist
Poudre Schol District R-1
2407 La Porte
Avenue
Fort Collins, CO 80521-2297

Introduction

In order to meet student needs for computer awareness and programming and to utilize computers for across-the-curriculum instruction and applications, schools need to plan for the placement of computers so that they can be used effectively for large or small group instruction or individualization.

When computers were first placed in the schools, the general configuration was for a single computer to be placed in each room. As more computers became available, some schools chose to design computer labs and place all of the computers in the laboratory. Other schools have their computers on carts and use them in individual rooms for part of the time and move them to a central location for large group instruction. However, most schools have had few sources of information about computer lab design. This paper covers design construction and equipment considerations that make the lab a better functioning facility. At Poudre School District R-1 in Fort Collins, Colorado, we use all of the described configurations for various learning environments.

Needs

At the **elementary** level, teachers indicate that they prefer alternative arrangements for computer use. In the primary grades one or more computers located in the classroom that can be used as learning centers seem to best meet student needs. For example, Writing to Read™ labs have seven learning centers with four IBM computers making up one of those centers. The Apple Learning Series: Early Education™ uses three computers clustered into a learning center. In order to teach keyboarding and word processing, a lab setting with enough computers for each student in the largest class or between 20 and 30 computers is preferred. To meet this need in schools that do not have adequate numbers of computers, we have a set of traveling computers that are moved from site to site and set up in the lab area or media center.

Schools which are designed around center pods sometimes prefer mini-labs in the pod area for one-third of a class, or nine to ten computers.

Media centers with electronic catalogues need computers located in their area. For this purpose we have designed carrels around central poles that contain wiring for the networked systems. Computers on carts with LCD screens are available for checkout to individual rooms.

A minimum ratio of one computer per 20 students allows one hour per student per week computer time.

Junior high schools have one lab in the business/keyboarding area and at least one general purpose lab. In addition mini-labs of ten to fifteen computer are planned for the major departments and the media center. These mini-labs may be made up of portable lap top computers. The media center has computers for the electronic circulation system, and checkout computers available for individual classroom use. The minimum ratio should be one computer per 15 students.

At the **high school** level, each building has full labs (25 - 30 machines) for writing, mathematics/programming, keyboarding/word processing, and vocational business classes; and has plans for mini-labs (5 to 20 machines) for computer-aided design, science, journalism, music and art. The media center has computers for the card catalogue, accessing databases and word processing. In addition, individual machines are available for checkout. The minimum ratio should be one computer per 10 students.

This assessment does not include computers for teacher and administrator needs, nor does it address the needs for interactive video or electronic teaching at a distance.

General Guidelines

The computer location and design specifications will be different at each different school based upon physical space available, teacher preference, instructional needs and grade levels. However, there are general specifications that can be followed in determining final design. Computer lab areas for specific disciplines such as science, art, or industrial technologies will have special equipment needs. These specifications will deal mainly with general purpose labs which will be referred to as computer resource centers or CRCs.

LOCATION. The general purpose CRC should be located as close to the media center as possible. The ideal location is a room that is immediately adjacent to or within the media center. A one-half glass partition which allows for visual monitoring of the CRC for the circulation desk is preferable. There are several advantages to this location:

*The software collection, cataloguing and check-out can be handled through the established media circulation system.

*The Computer Resource Center fits into the concept of a media learning center where students and teachers come for all support materials to supplement teaching and learning.

*In times of low use the CRC can be monitored by the person monitoring the use of the media center. If the school prefers to shut the CRC off from the media center visually, they can do so with miniblinds or shades.

*The partitions between the CRC and media center will limit noise problems.

Mini-labs need be visible to teachers whose students will be using them so should located in areas within or immediately adjacent to classrooms.

For security reasons no external doors should lead into the CRC. Doors that do lead into the center should be capable of being locked with a dead-bolt lock. A fire door is preferable. Outside windows should be eliminated if possible. If not they should have lockable metal louvre systems installed. Transoms over internal doors should be secured.

Although the area must be secure, it should be easily accessible so computer centers can be opened at times that the whole school is not opened so the facility can be used by parents or the general community if the district so desires. In our new school design, we attempt to have a learning resource wing which can be secured from the rest of the building and accessible from the outside much like the gymnasium wing.

***DESIGN*:** Several basic designs can be adapted for use in individual schools:

Perimeter Design *(See Illustration A)* In this design the computers are located round the outside edges of the CRC. Some advantages of this design are

*More computers can o[illegible]e placed in a smaller area.

*The large center area can be used for an additional bank of computers to be added as more work stations are needed, *(See Illustration B)* or for work tables.

*Mobile computer carts or regular tables can be used rather than specially constructed tables containing space for electrical wiring.

*Wiring can be installed on the walls behind the computers.

*The teacher can see all of the computer screens from the center of the room to monitor in what activities the students are involved and can move easily from student to student.

*The traffic pattern into the room does not require students to walk around computers or wiring.

Some disadvantages of this design are:

*When the teacher is giving instruction, the students will either have their back to the instructor or will have to turn physically from their computers to face the teacher. Although this disadvantage can sometimes be an advantage because it forces attention on the teacher and away from the computer and gets hands off the keyboard, the student does not have a desk area on which to write when turned away. A pull-out work area (writing arm) might help in this situation. (Actually, teachers should not be lecturing in a computer center.)

*The perimeter design limits easily accessible storage, bookcase, writing board and bulletin board space along the walls.

Row Forward Design: *(see Illustration C)* In the row-forward design, computers are placed on long tables so that the students are all facing the front of the room where the teacher has a teaching station.

The advantages of this design are

*The teacher can lecture or give directions to the students without having them turn around, and the students can use the table space next to their computers for a writing space.

*The students can copy materials from the board or from a central monitor directly onto their machine.

*Wall areas can be used for writing and bulletin boards and storage areas.

*The students can attend to both the teacher and their computer monitor. This would be advantageous in classes where students are learning to run an application program or learning programming by modeling the teacher's instruction, as opposed to classes where the students are using the computer as a tool.

The disadvantages of this design are

*The students will tend to disrupt others as they move in and out of the aisles to their computers.

*Special tables need to be constructed to contain the wiring and/or networking for the computers.

*The teacher cannot easily monitor all of the machines from one location, although they may be able to see most screens from the back of the room.

Mini-labs. These labs should be seen as an integral part of a classroom and should be located in a bank across one wall of the classroom or in a pod area immediately adjacent to the room. Generally, a mini-lab should be on moveable carts so the teacher can readjust the room arrangement to meet teaching needs. Power poles with adequate plug-ins can be used as "hitching posts" for the mobile carts. These posts can contain network wiring in addition to electrical wiring.

Carrel Arrangement *(see Illustration D).* This arrange-ment allows the student to have some privacy and isolation while working on the computer. It would not be effective for teaching in a lab setting but might be desirable for a media center or word processing area.

The CRC should be large enough for the largest class in the school. Thirty square feet per child is a good "rule of thumb" to follow in allowing space for the CRC. Thus, most CRCs should be 900 square feet.

Cool, neutral colors are best in the computer center, although graphic designs in primary colors that are not distracting are pleasing.

Flooring should produce minimum static electricity and be aesthetically pleasing. Linoleum or tile is the most static free. However, several types of carpet are quite satisfactory. The static electric level should be below 2 KV. Armstrong's Deliberation TEC (Total Electric Compatibility) and BASF Zeftron Nylon Antishock, Zeftron Nylon Sefstat, and Zeftron Nylon Conductive carpets are several types that will withstand different levels of static electricity. Oelefin carpeting with low KV ratings may also be used. If you must use existing carpeting in the CRC, treat the carpet once a week with at static guard carpet spray. These can be purchased commercially or made from a mixture of one part fabric softener or 20 parts water. On hard floors, it is important not to have a wax build up since wax is a source of static electricity.

Do not have chalk boards or pencil sharpeners in the CRC. You may use the dry erase boards or an overhead projector for instructional purposes.

There should be bulletin boards in the lab for information about the lab use.

***SAFETY*:** Since computer areas have a concentration of electric equipment and wiring, safety precautions are very important.

There should be two doors into the computer center and the doors should swing out. The aisles that proved a student"s means of exit must be at least five feet wide if there are two rows of chairs backing on them. All construction materials in the room should be flame resistant. Fire extinguishers, preferably Halon-type, which smother electrical fires should be located in all computer centers.

***WIRING*:** Wiring must be adequate to support the increased demand. There needs to be an electrical circuit for all the instructor's equipment that is separate from that of the students' so that machine can be turned on independent of theirs.

There should be a master switch which controls power to all student computers and a separate master switch that controls the second plug in the four-plug outlet. By separating the circuitry to the station, the teacher can choose to plug the monitors into one plug in the four plug set and the computer into the other. Thus, they can switch off all student monitors without turning off the computer. This is a good tool to use when providing instruction.

There should be one circuit for each ten computer work stations.

The circuitry for the overhead lights should be independent of the computer circuit system.

Each individual workstation should be wired into an outlet box with a master on/off switch channeled through a circuit breaker for protection. Thus, the component parts can all be turned on with one switch. There should be at least four plug-ins for each unit.

Floor plugs should not be used.

A video distribution system should be installed in labs where classroom instruction will take place. This system allows the students to switch their monitors to see what is on the instructor's monitor. *(See Illustration E)* More sophisticated systems allow the teacher select and distribute the system from any monitor in the center. LCD screens that attach to the instructor's computer may be used in place of a VDS to project the screen image.

When new centers are constructed, wiring raceways for potential networking should be installed. Since requirements for the type of wiring needed will be based upon the networking system purchased, raceways should have pull-wires to facilitate the changing wiring needs.

A local area network (LAN) for the entire building will facilitate instruction. Students can access materials on the hard disk, including the media catalogue, from throughout the building. We use Corvus Omni-Networks with twisted pair wiring and AppleTalk with PhoneNet for building-wide and lab LANs.

If the CRC is not networked with print spooling, then a print queuing system or switch boxes should be installed.

A telephone line should go into the computer center for telecommunication uses. This should be a dedicated line. The connector box should be located near the teacher's station.

***ENVIRONMENTAL FEATURES*:** Sound control is important because equipment such as printers and computer fans produce noise. Acoustical materials used on the ceilings and walls and carpeting on the floors will reduce the general noise levels. If high speed, letter-quality printers are used, they should be encased in acoustical boxes. Computer program should have the option of turning the sound off. Rheostats can be installed on computers that do not have sound control to regulate the volume. Earphones should be proved when computer programs require sound.

Lighting is also an important consideration because it must be adequate but must not produce glare on the screen. A diffuser cover over two-light fluorescent bulbs seems to provide the best, non-glare light.

The temperature in the CRC should be maintained in the 65-75 degrees Fahrenheit range. Since most CRCs are located in interior rooms with no window access, it is important that some type of ventilation or cooling system be considered. These rooms get very warm and close with 25 students working on heat-producing computers. A ventilation system that filters dust will help decrease maintenance on the computers. Air conditioning is preferable (swamp coolers can be used in dry climates but are not as satisfactory) but strict temperature control is not as necessary as it was when computers were more temperamental.

A humidity range of 30 to 50 degrees is ideal for computers and to lower the static electricity problem. This is difficult to acquire or maintain in our dry Colorado climate.

***FURNISHINGS*:** There need to be locked storage cabinets for disks and supplies that are to be kept in the CRC. At least one cabinet must be low enough so that heavy boxes of paper do not have to be lifted over tables to a wall cabinet. Open shelving for software should line the walls. If a networked, there most be a space designed for the file server that is accessible by the computer center manager but not by students.

Individual stands for computers can be used and have the advantage of being easily transportable if the school needs to use computers from the resource center in classrooms. There are a variety of these stands. Be sure to allow enough space on the stand for a book or paper and printer.

Poudre District R-1 builds its own tables for most district computer resource centers. These tables include a channel to house the wiring. Ideally, 48 inches should be allowed per workstation if students will ever work two to a machine. Three feet should be allowed if all stations will be used by single users. Approximately three feet should be allowed between tables in the row-forward design. Five to six feet will be necessary when students are back to back. The height of the chairs and tables should vary for the different grades. At the junior high and high school level, table height is usually 25-26 inches; elementary is 24 inches. The width of the table may need to be altered to fit the footprint of the computers being used in the computer center. If Apple computers are being used, 24 inches is adequate. For IBM and Tandy computers, 26 inches should be allowed.

For the instructor station, there should be a desk, a computer stand with room for a printer, and a file cabinet. A large screen monitor and VCR for interactive video may be desirable.

Computers used in the media center for electronic circulation system should be on higher tables that allow the user to stand to operate them.

***EQUIPMENT*:** Each student station should have a computer with a single disk drive and a green screen monitor as the minimum requirements. At the elementary and junior high level and in art and industrial science departments in the high school, the monitors should be color. At the junior high ahd high school level, the stations should either be networked to a file server or have a second disk drive. All new Apple computers must have at least 128k of memory. Apple *IIgs* machines should have a megabyte of memory. MS-DOS machines should have a minimum of 512k.

Minimumly, there should be one printer for every five computers with some type of switch, queuing, or print spooling device. If the CRC is used extensively for word processing or other hard copy production, the printer/computer ratio should be reduced or a high speed printer used.

When large class instruction is done, all of the machines should be of the same model. Having different machines available in the building or in centers that are used for individualized instruction or applications does not present a problem and can actually be an advantage in teaching computer concepts.

The teacher's station should have a dual disk drive; a monitor with color/green screen capability; a printer; a large screen monitor, LCD display and overhead projector or video distribution controls; and a modem.

• • • •

These are the basic specifications for construction of computer centers in schools. Configuation of the stations and lab design should be flexible to meet the neeeds of teachers and students and the changing technologies. Although we carefully design our computer centers, we are mindful that as technology and instructional delivery styles change so will our facility needs. While a computer resource center seems to work best now, the future may bring us to "the classroom of one," and centers may become obsolete. Even now, we provide a multitude of alternatives to computer centers -- CD-ROM learning stations in the media center, Interactive laser disk stations in classrooms and desktop publishing stations in work areas for example. As with any educational design, learning must dictate facility planning.

The criteria put forth in this paper represents the results of careful study, work with architects and maintenance personnel, and observation in the 49 computer centers in Poudre School District R-1 in Fort Collins, Colorado.

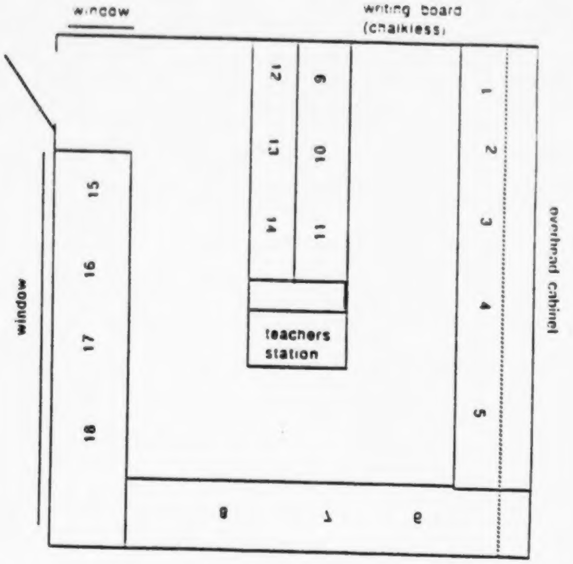

window
writing board (chalkless)
window
overhead cabinet
teachers station
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

Illustration A

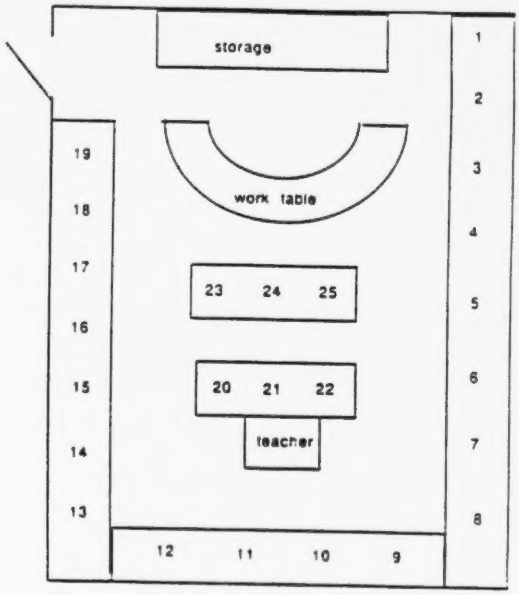

storage
work table
teacher
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

Illustration B

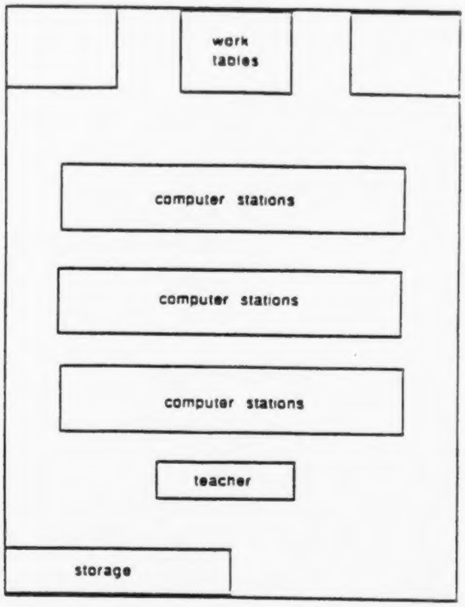

work tables
computer stations
computer stations
computer stations
teacher
storage

Illustration C

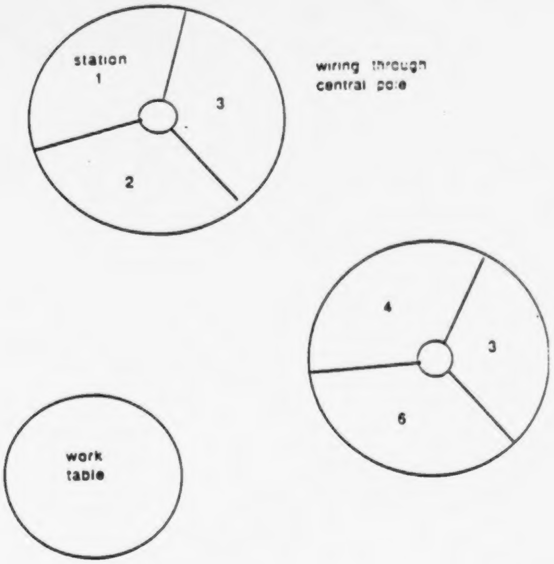

station
1
3
2
wiring through
central pole
4
3
6
work
table

Illustration D

COMPUTER VIDEO DISTRIBUTION SYSTEM

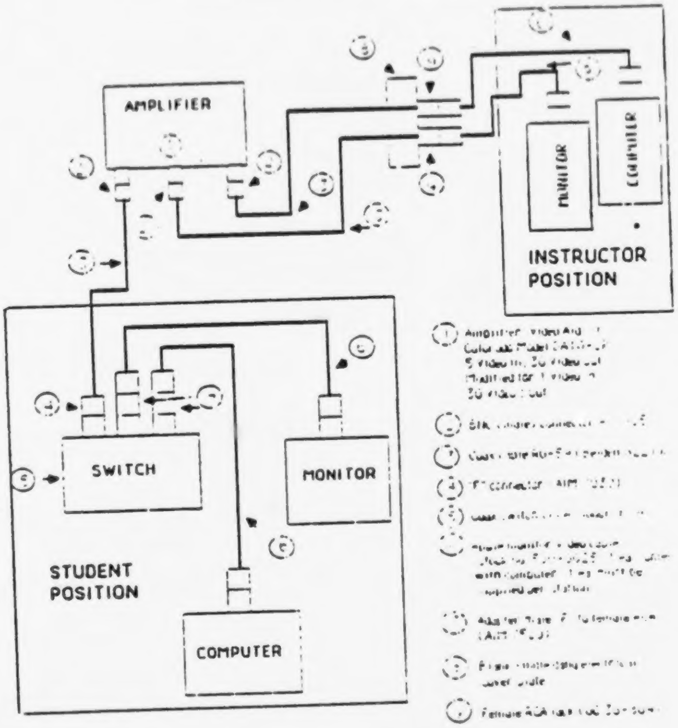

AMPLIFIER
MONITOR
COMPUTER
INSTRUCTOR
POSITION
SWITCH
MONITOR
STUDENT
POSITION
COMPUTER
Amplifier Video Aid
5 video in, 30 video out
Modified for 1 video in
30 video out
Female RCA

Illustration E

EDUCATIONAL COMPUTING IN THE USSR

Barbara Kurshan, Chair
Educorp Consultants
Roanoke, VA

ABSTRACT

Exploring educational computing in the Soviet Union is a unique experience for American educators. A review of the current state of Soviet educational computing evokes the eerie feeling of *déjà vu*. Upon visiting the Soviet Union you see educational computing as it was ten years ago in America. This panel will discuss the following topics as experienced by the panel members on recent trips to the Soviet Union:

- Current state of USSR educational computing;
- Future directions for educational computing in the USSR;
- Cooperative US and USSR computer education projects; and
- Special computer education projects:
    - Microcomputers in the USSR
    - Telecommunications
    - USIA Information USA Exhibit

PANELISTS

Rachelle Heller
George Washington University
Washington, DC

Karen Billings
Claris Corporation
Mountain View, CA

Peter Copen
Copen Foundation
Yorktown Heights, NY

Lynne Schrum
MIX MOderator
Eugene, OR

Guest from the Soviet Union

SPONSOR: ICCE and SIGCUE

THE REQUIRED COMPUTER COURSE FOR EDUCATION MAJORS: A NATIONAL PERSPECTIVE

Pamela Moore, *Chair*
Eastern Michigan University
Ypsilanti, MI

ABSTRACT

In June 1987, at NECC '87, The Special Interest Group on Teacher Education of the ICCE established the Required Computing Course Committee. Its mission had two major thrusts: 1) to survey a national sampling of teacher education institutions to discover what was included in the computer education course required of all teacher candidates; and 2) to predict national trends in the directions the course was taking.

The committee decided on a general plan of attack that has resulted in two years of study. The first year of study focused on collection of information from several states on: 1) the current status of computer education courses, certifications, and degree programs; and 2) the development and refinement of the survey instrument for use in collecting the information nationwide. A report of the first year's work was reported at NECC '88.

The second year of study focused on the survey of a nationwide sample, analysis of the data, and preparation of a final committee report. The report includes a summary that identifies the course content, target audience, facility, instructional materials, class activities, methods, and future plans for required computing courses taught at preservice teacher training institutions around the nation.

The discussion will include the following:

1. The results of the survey regarding:
   a. The descriptions of required computer education courses;
   b. National and regional trends in the development and refinement of computer education courses;
   c. Future implications on computer education for teachers.
2. An evaluation of the results of the survey and suggestions for further study.

PANELISTS

Joyce S. Friske
Oklahoma State University
Stillwater, OK

Mildred D. Lintner
Eastern Michigan University
Ypsilanti, MI

Bonnie Mathies
Wright State University
Dayton, OH

Charles Mlynarczyk
State University College
Potsdam, NY

Lajeane Thomas
Louisiana Technology University
Ruston, LA

James Wiebe
California State University
Los Angeles, CA

SPONSOR: ICCE

REAL TIME COMPOSITION:
TEACHING WRITING WITH A LOCAL AREA NETWORK

George H. Brett II, *Chair*
The University of North Carolina
Research Triangle Park, NC

ABSTRACT

Teaching writing with computers has been a source of great activity in recent years. There has been much discussion about methods and merits of teaching writing this way. Faculty have written computer programs or used commercial word processing packages to teach writing and composition. However, until recently the combination of computer and word processing program has been used more as an 'electric typewriter' by students. The 'electric typewriter' is often used for pre-writing exercises which evolve into final papers that are printed out on paper.

This session will be a demonstration of two interactive writing programs. These systems depart from the standard procedure in that they use microcomputers, a local area network, and software to teach writing and composition in a real-time environment. Because the computers are all linked by a local area network, students and teacher are able to share papers electronically. The end result is an interactive process of writing where a teacher is able to insert his or her comments directly into the student's writing while the student and class are watching.

The session will have a live demonstration of two interactive or responsive writing networks. One system, using MS-DOS microcomputers is currently used at Gallaudet University in Washington D.C. The other network program, using Apple Macintoshes, has been developed and used at the University of South Carolina at Columbia, South Carolina.

PRESENTERS

Trent Batson
Director, ENFI Project
Gallaudet University
Washington D.C.

Robert Cannon
Computer Science Dept.
University of South Carolina
Columbia, SC

SPONSOR: ACH

ACOT ROUNDTABLE DISCUSSION

David Dwyer
Apple Classroom of Tomorrow (ACOT)
Apple Computer, Inc.
20525 Mariani Avenue, MS: 42C
Cupertino, CA 95014

ABSTRACT

Two ACOT teachers, Dwayne Marshall and Sue Misiak, will be discussing how they use a variety of interactive technologies with high school students in Columbus, Ohio.

West High School is the only secondary school in the ACOT project and the only site that uses primarily Macintosh-based technology. Since very little pre-college instructional software has been written for the Macintosh, teachers have had to rely on word processing, database, and other productivity software to build a high school curriculum. In addition, they are using HyperCard, interactive video, scanners, MIDIs, and robotics technologies to develop their own learning tools.

English/social studies and math/science curricula are currently tied together in an interdisciplinary approach. This arrangement helps facilitate one of the site goals which is to develop an interdisciplinary curriculum that relies on team teaching and high access to several technologies. A math and science project completed last year involved the use of robotics technology to construct a model of downtown Columbus. The model included accurately scaled buildings within a city environment that was controlled by eleven Macintosh computers. Robotics included cars, traffic lights, city lights, an elevator, a railway crossing, and a construction site with crane and steel post driver.

This year, students studied the Constitution and the Bill of Rights in their American History class. In American Literature they researched individual amendments and court cases, and recorded their reports and individual responses on HyperCard stacks enhanced by scanned, original, and clip art. Additional technical applications included the use of the MacRecorder, animation software, and specialized color software. This classroom assignment eventually evolved into a multi-media presentation for the Ohio Technology Fair. The presentation included student stacks, a videotape, scanned colorized images, and MIDI-produced musical tracks.

The Columbus site has extended the team-teaching approach by including students and outside resource people in additional interdisciplinary curricula. They are developing a combined art and music program and expanding the cross-curriculum publishing and multimedia projects, such as *The Bill of Rights* project. Teachers are also developing more student-centered activities and curriculum units. For example, using telecommunications, the high school students will serve as mentors for younger ACOT colleagues in other cities.

TECHNOLOGY IN ELEMENTARY SCHOOLS

Don Rawitsch
Victor R. Delclos
Beverly Farr, Chris Franklin

ABSTRACT: The Effect of Computer Use and Student Work Style on Database Analysis Activities in the Social Studies

Don Rawitsch, Administrative Vice-President, MECC, 3490 Lexington Avenue North, St. Paul, Minnesota, 55126.

This presentation describes a study investigating the effects of using computer databases in junior high social studies learning activities. The study involved 339 eighth graders from three Minnesota schools. The study evaluated students' problem-solving capability using computerized databases versus traditional methods of analyzing data. It also assessed student work styles and attitudinal effects of computer use during lessons in world geography. Four hypotheses were explored:

- Students will be able to solve problems requiring the comparative analysis of data more efficiently by using a computerized database as compared to non-computer methods.
- Students will have more positive affect toward computerized methods of solving problems requiring comparative analysis of data than towards non-computer methods.
- Students with a structured work style will be more efficient in computerized database use than students with an unstructured work style.
- Students will be able to apply computerized database methods learned in social studies related to activities that more closely relate to future employment and/or home environments.

ABSTRACT: Mediated Instruction and Logo: Examples and Effects

Victor R. Delclos, Tulane University, Department of Education, New Orleans, LA 70118.

This presentation will use videotaped segments of instruction and videotaped real-time screen dumps of students completing Logo projects to demonstrate examples of two approaches to Logo instruction and to offer evidence of the different effects of those approaches on the problem-solving performance of fifth and sixth grade students. The first approach to instruction focuses on "bridging"—explicitily making connections between newly encountered problems and relevant solutions encountered earlier in training. The second approach focuses on the explicit discussion and application of general problem-solving strategies such as decomposing the initial problem and debugging errors.

The overall goal of the research project from which these examples are taken is to evaluate the unique contribution of various specific instructional components to a general approach called "mediated instruction" Computer output recordings will show examples of the different consequences of each of these instructional components on students' efforts to sovle new problems involving Logo turtle graphics.

ABSTRACT: Research Design for Evaluation of Model Applications of Computer Technology in Indiana Classrooms

Beverly Farr, Advanced Technology, Inc., 2601 Fortune Circle E., Indianapolis, IN 46241; Chris Franklin, Indiana Dept. of Education.

Purpose of the Study:

To evaluate a variety of computer usage categories as they are implemented in 4th grade classrooms. The study will use a combination of an inferential approach to determine the size of the difference that computers can make on achievement with ethnographic methodology to reveal in as rich detail as possible the characteristics of the teaching-learning process in classrooms using computer technology.

Since the use of computers in schools can vary substantially, a study of effective use must take into consideration a variety of factors. Criteria used to define usage categories include: setting, ratio of students to computers, format of use, subject area(s) addressed, type of software program(s), time spent at computer, use for demonstration.

Questions underlying study:

1. What are the effects of computer use on students?
2. What are the effects of computer use on teachers?

3. What are the curriculum implications of the use of computers?
4. What logistical factors contribute to the effective use of computers in a school?

Methodology:

Usage categories were defined which combined various criteria from those identified above. These include:

1. One computer/classroom
2. Computer lab
3. Self-contained classroom (approx. 12 computers in a classroom)
4. Mini-labs (2 types of hardware; 6 computers on movable carts travel among 4th grade classrooms, each one having access for 1/4 of school year)
5. WICAT lab
6. Take-home project (5 sites with Apple or IBM hardware; all 4th graders are given a computer to use at home)

Evaluation includes two site visits during the school year (classroom observation, interviews, student and teacher surveys). Criterion measures include standardized achievement test data available on experimental and control classrooms, criterion tests designed to test curricular-related activities taught with computer, and attitude surveys.

IMPLEMENTATION STRATEGIES

Patrick C.J. Nolan, David H. McKinnon, Robert R. Sharp
Michael-K. Russ, Kim Brown

ABSTRACT: Curriculum Integration with Computers in New Zealand Secondary Education: Professional Development Support for Teachers

David H. McKinnon, Patrick C.J. Nolan, Robert R. Sharp, Education Department, Massey University, Palmerston North, New Zealand.

Studies of curriculum innovation have demonstrated that the teacher is a key factor in the process of successful innovation adoption. This paper examines the role of the teacher in a three year curriculum research and development project presently being conducted in a New Zealand secondary school, documents the innovation adoption process that has occurred, describes the concerns of teachers regarding the three main elements of the project (computer use, subject integration, and out of class activities), and reports on strategies that have been used in the construction of a professional development support program.

ABSTRACT: Curriculum Integration with Computers in New Zealand Secondary Education: A Research and Development Project

Patrick C.J. Nolan, David H. McKinnon, Robert R. Sharp, Education Department, Massey University, Palmerston North, New Zealand.

This paper describes the design, purpose, and structure of a New Zealand secondary schools Integrated Curriculum Project, examines the implementation strategy, and reports on student attitudes toward the three principal elements of the project—the application of computers, involvement in out-of-class activities, and the integrated study of traditionally compartmentalised school subjects. The Attitudinal aspect involves a comparison between two similar cohorts of students. One of the cohorts is enrolled in integrated studies programS while the other is participating in conventional school programs.

Collaboration among teachers in planning, preparation, and shared teaching appears to be the key to effective integration of subjects. The use of the computer as a learning tool supports subject integration, and out-of-class activities appears to motivate students and helps give their school learning a clear direction, meaning, and purpose.

ABSTRACT: Extending Study Opportunities With Portable Computers

Michael K. Russ, Kim Brown, Evansville-Vanderburgh School Corporation, 1 S.E. 9th Street, Evansville, IN 47708.

Students, teachers, and parents are all essential for effective learning. A lack of participation by any one of the three will result in lower student performance. A large number of studies indicate that parent involvement of any kind improves student achievement. This project involves direct parent participation. Parents must attend a workshop at school in order for their child to checkout a computer for home use. Parents are encouraged to involve themselves in the child's computer work at home.

This project, funded by a grant from the Indiana Consortium for Computer and High Education Technology, was set up to link school learning opportunities with home learning opportunities. Fifth grade students at Lodge Elementary School take the computers home two times a week. The computers are powered by batteries, though AC power packs are available.

The goal of the project is to allow both teachers and parents to work together with the same equipment and materials to provide effective learning opportunities for children. A variety of software is used to accomplish this. Included is word processing, LogoWriter, keyboarding, and CAI software in math, reading, and language arts.

ENRICHMENT OPPORTUNITIES FOR AT-RISK YOUTH

Antonia Stone, Chair
Executive Director
Playing To Win, Inc
New York, NY

ABSTRACT

Are we on a disaster course by assigning extra remediation hours to our "at-risk" young people? Are we depriving them of the enrichment activities offered to their less risky peers—opportunities for project work, for the cultivation of thinking skills, for discovery in music, for networking with other students across the country?

Each panelist will describe a different project designed to bring enrichment opportunities to "at-risk" young people:

- Higher Level Thinking Skills: Carol Edwards
- Music, Electronic and Otherwise: Lylburn Dowling
- Telecommunications with TeenNet: Seth Chaiklin

Time will be allowed for questions and comment from those attending.

PANELISTS

Carol Edwards
Southern Coalition fro Educational Equity
Atlanta, GA

Lylburn Dowling
Musician
New York, NY

Seth Chaiklin
Columbia University
New York, NY

SPONSOR: ICCE

CAN COMPUTERS TEACH? LESSONS FROM 25 YEARS OF RESEARCH

Jerome Johnston
The University of Michigan
Ann Arbor, MI

ABSTRACT

In the midst of the great rush to computerize every facet of the educational enterprise we need to ask what is known about the effects of computer-based learning. This presentation conceptualizes the various roles that the computer can play in education (tutor, tutee, tool, instructional monitor) and describes the key research findings concerning the impact of computers used in these ways. Emphasis is placed on the conditions necessary to yield desired results, with a particular emphasis on the roles that teachers must play. This talk is adapted from Johnston's book, *Electronic Learning: From Audiotape to Videodisc* (Erlbaum, 1987) with additions from more recent research findings.

SPONSOR: SIGCUE

IT'S NOT TOO LATE TO TAKE A "SHOP CLASS": LEARNING SYNTHESIS SKILLS THROUGH PROGRAMMING

Elliot Soloway
Department of EECS
University of Michigan
Ann Arbor, Michigan 48109

ABSTRACT

By and large school education stresses "analysis skills": analyzing a story, poem, map, etc. However, in the real world, people need to act—and thus they need to develop "synthesis skills", too. In the old days, kids in "shop class" actually got a chance to explore the composition of and interaction among various components, thereby developing synthesis skills. Programming has the potential for being the new shop class: it is not as mired in history, as is the teaching of English, mathematics, etc., and it definitely is a "doing" subject. Based on our studies of software experts we have developed a characterization of what they do that is teachable—to just plain high school kids. To support that novel curriculum, we are developing a "SUPERstructure" editing environment in which students can operate on multiple, non-isomorphic representations of the system under construction: the requirements (goals), the methods (plans), and the implementation (code). In addition to highlighting our classroom experiences, we will speculate on mechanisms for transforming the classroom via theory-informed technology.

LANGUAGE: THE NEXT GENERATION OF WRITING TOOLS

Midian Kurland
Bank Street Software
55 Wheeler Street
Cambridge, MA 02138

Art Bardige
Learningways, Inc.

ABSTRACT

Word processors that have been the mainstay of educational computing for nearly ten years do little to actually help students learn to write. They were designed to be editors for programmers and automatic typewriters for secretaries. A new generation of *Process Writers*—computer tools for writers that actually can help people collect and organize information and ideas, manipulate drafts, and produce pretty results—are just now becoming available. Some Process Writers can even stimulate a student's thinking. We believe that this new generation of tools that reflect the writing process will actually allow us to teach students to write better. In this presentation, Midian Kurland, Director of Software Development for the Bank Street Software Group, and Art Bardige, President of Learningways, Inc., will present their latest process writing programs and describe how they foresee writing tools *and* writing curricula evolving in the future.

ELEMENTARY SCIENCE AND MATHEMATICS

Dennis Kurtz, Peter Connors
Kathleen Maury
Michael J. Rush, Peggy Buchanan, Mark Bloede

ABSTRACT: Project STEP (Space Technology Environment Program)

Dennis Kurtz, Peter Connors, Half Hollow Hills Schools Planetarium, 50 Vanderbilt Parkway, Dix Hills, NY 11746.

Project STEP, Space Technology Environmental Program, is located in the Half Hollow Hills School District, Dix Hills, NY. The project is centered around a forty foot by ten foot mock-up of the US Space Shuttle. The facility consists of a functioning orbiter flight deck, a science lab, and a working mission control. The STEP facility provides an excellent stage for simulations.

The space shuttle flight simulations are hands-on, open ended learning experiences, which cannot be compared in value to traditional learning methods. This is due not solely to the subject matter, but also because of the manner in which it is taught.

Project STEP uses the lure of space travel to motivate children to learn complex ideas, think critically, and work as a team. Students are exposed to similar disciplines as real astronauts and mission specialists. All students learn to navigate our shuttle, communicate, and perform experiments.

Cooperative effort is our main theme. Participating students, whether they work in mission control or with the flight crew, leave with a new understanding of responsibility, real-time decision making, and planning.

ABSTRACT: The Effects of Calculators and Computers on the Mathematics Curriculum

Kathleen Maury, University of Wisconsin, 1214 North 20th Street, Superior, WI 54880.

When calculators and computers are available in classrooms, they often effect the content of the mathematics curriculum. Research indicates that in technology-rich environments, mathematical skills which lend themselves to machine execution become devalued. In these classrooms, machines assume the role of executing mathematical algorithms, and the humans spend their energies on other tasks.

Many believe the idea of the effective procedure may be the most important idea of the 20th century. However, it may be important for reasons that are not well understood and also are well beyond the obvious. The effective procedure (because it is machine executable) may actually provide educators with the power to choose the mathematics that is worth human energy. Research indicates that school personnel will devalue those algorithms that can be written as effective procedures, stored as effective procedures, and made accessible to students and teachers in school. Teachers then decide that these algorithms are no longer worth human energy, and drop or deemphasize them from the school mathematics curriculum. In their place, teachers substitute other topics. In effect, humans have developed effective procedures and machines that will store and execute these procedures, and are spending less instructional time teaching humans to execute them. Instead, instructional time is devoted to other things.

If these trends continue, we may see a mathematics curriculum quite different from the traditional one we've come to know. The change may not reside in the quality, the quantity, or the style of teaching or learning. The change may not reside in the things of schools, the tools, or the general appearance of schools or classrooms. However, the people may be doing different things, the nature of what they are doing may be different, and the purpose of what they are doing may change. The most important change wrought by machines may lie in what the people choose to do with their energies and time, along with the definition of and the value placed on human intellegence.

ABSTRACT: Integrating Telecommunications into Middle School Science

Michael J. Rush, Peggy Buchanan, Mark Bloede, Project INSITE, Zionsville, IN 46077.

Project INSITE is a cooperative effort of eight Indiana school districts to develop a comprehensive training program which effectively incorporates the use of telecommunications into the middle school science. Telecommunications, as defined by this project, includes: online searching, teleconferencing, electronic bulletin boards, and electronic mail. The two year project is funded by the National Science Foundation under its Private Sector Partnerships to Improve K-12 Science and Mathematics Education. Teachers, administrators, media specialists, and faculty from Purdue University and Ball State University worked with scientists from Eli Lilly and Company and Merrell Dow Research Facility to create six instructional units pertaining to current issues and concerns and focusing on hands-on, non-textbook oriented activities. The units integrate telecommunications into the activities to provide students access to information not available in their schools, allow them to communicate electronically with the scientists, and participate in interdistrict experiments. The session will include a description of the project, a discussion of hardware, software, budgetary considerations, successes and failures, and include a demonstration of online searching. Examples of student projects and samples of materials will be distributed.

PRACTICAL GUIDELINES: USING TELECOMMUNICATIONS IN THE CLASSROOM

Lynne Schrum
University of Oregon

Len Scrogan
Princeton Regional School District

Cecilia Lenk
Technology Education Research Center

ABSTRACT

This presentation focuses on five unique characteristics of telecommunications that will help teachers design and implement 'on-line' classroom experiences. Unless the curriculum remains the essential goal, however, telecommunications will only become a quick, not-so-cheap thrill and will fade away. Examples from unusual recent student demonstration projects are given.

First, telecommunications is timely; given a specific goal, communications can be carried out instantly. During the presidential campaign, it was difficult to keep current on the candidate's opinions and to excite students about the election process. However, with the speed afforded via modem and computer, some classes had the opportunity to communicate directly with the candidate's offices and have their questions answered directly.

Second, in most classrooms one teacher and 30 students work on a project. Using the storage capabilities built into telecommunications networks, multiple participants can be actively involved. Teacher and student planning can take place; more importantly, data can be posted once and then shared by all involved classes. This occurred in a plant growing contest where schools around the country started with the same seeds and compared the growing conditions and weekly progress of their plants.

Next, through the use of electronic networks, experts can be brought right into a classroom. In one school, a poet was 'in residence' to a high school composition class, while maintaining his usual work schedule at his home three thousand miles away.

Further, by actively participating in self directed research, many classes can begin to discover local, state, and national issues, compare and contrast the efforts of their own communities and others, and begin to synthesize and evaluate possible solutions. This model is currently being tried in a project on the 'Plight of the Homeless' issue in schools across the continent.

Last, using telecommunications, any class can connect with other cultures, languages, and environments from all over the world. It is now possible for a Spanish class in Omaha to create a newspaper with classes of native Spanish speakers in Puerto Rico, to find out about the language they are learning, and to gain an appreciation for another country's history and traditions.

The kinds of activities that lend themselves to the special attributes of telecommunications are limited only by your imagination—or that of your students.

TECHNOLOGY IN ART EDUCATION

Kathy Rehfield, Karen Warner
Harold J. McWhinnie
David Alexander

ABSTRACT: Videodiscs and Hypercard in Art Education

Kathy Rehfield, Computer Coordinator, White Plains Public Schools, 500 North Street, White Plains, NY 10605; Karen Warner, Regional Curriculum Services Unit - Central.

Our use of videodiscs and Hypercard stacks with high school art teachers has proven the strong impact that this technology brings to art education. In essence simulating an interactive visit to the great museums of the world, videodiscs bring quality visual reproductions of art work into the classroom. The control afforded by Hypercard programming allows teachers to send students through multiple museum tours with differing goals, perspectives, viewpoints and objectives. Our project included demonstration of the National Gallery of Art, Vincent van Gogh, and the Louvre videodiscs and their accompanying Hypercard Laserguides by the Voyager Company to high school art teachers. The Macintosh and videodisc player were then made available to the teachers who used the technology in their lessons. Diverse uses and implementations emerged which took special advantage of the branching and interactive nature of computer control. Some specific interactive activities which were programmed through the Hypercard Laserguide include: having students match text-based descriptions of works of art with their visual reproduction; matching the period of a work of art to its picture; matching artists with their works; finding the odd picture out in a group of 5 paintings, etc. The presentation will include demonstration of the system, discussion of training and implementation models, and sharing of teacher generated lessons that take advantage of the technology.

ABSTRACT: Aesthetic Analysis of Selected Works of Van Gogh

Harold McWhinnie, University of Maryland, Department of Design, College Park, MD 20742.

This paper will present some research which makes use of the computer with a videodisc player to do aesthetic analyses of the work of Vincent van Gogh. The paper will consist of:

(a) a discussion of videodisc systems;

(b) a discussion of several software systems:

Dr Halo;
Handy;
PC-Paint;
Pilot.

This paper presents only several possible uses of the computer with interactive video disks for aesthetic analyses and creative efforts. It will also present the brief results of a pilot study undertaken in the Fall 1988 Semester.

ABSTRACT: What Can Be Said Developmentally About Young Children's Computer Drawings?

David Alexander, Tufts University, Eliot-Pearson Department of Child Study, Curriculum Resource Lab, Medford, MA 02155.

This study began as an attempt to determine the appropriateness of single keystroke computer drawing for young children (4-6 years) in a classroom setting. It continued in order to determine any developmental significance for children doing computer drawing. For children who came to the computer often enough to develop a drawing style, developmental progression became noteworthy.

Children's computer drawings qualitatively and developmentally match the drawings children do on paper, in so far as the line, color, and shape in both mediums are controlled by the artist to create certain effects.

An analysis of children's computer drawings, when the sample was large enough for a given child, showed qualitative and stage-like development with the qualities and properties of early stages found embedded in next and subsequent stages.

Children appear to accept, evade, or negate the constraints of computer drawing software to make drawing a creative, artistic, and developmental endeavor.

MODELS OF POLITICAL EFFECTIVENESS AT STATE AND LOCAL LEVELS

Bobby Goodson, Moderator
Bobby Goodson & Associates
1666 Kitchener Drive
Sunnyvale, CA 94087

ABSTRACT

Over the past few years educators have learned, very often through hard and painful lessons, that they need to develop new skills if they are to establish and maintain the technology projects in which they believe. It is no longer sufficient to get the training you need to develop your own skills, to think through an idea well enough to write an exciting proposal describing your idea, and put your ideas to the test in your own classroom. Teachers are learning, out of necessity, to find funding, write proposals, sell their ideas, and fight for their existence.

Panel members, speaking from their experience at state and local levels, will discuss a variety of efforts to obtain funding, influence policy, and further the successful implementation of educational technology projects. They will present both successful and unsuccessful models, and discuss the common elements and the lessons learned. Questions from the audience will be addressed and suggestions made in light of past experience.

PANELISTS

Sheila Cory, Coordinator
Chapel Hill, North Carolina

LeRoy Finkel
Educational Technology Committee, California

Geoffrey Fletcher
Texas Education Agency
Austin, Texas

THE IMPACT OF INFORMATION TECHNOLOGY ON THE ROLE OF THE SECONDARY SCHOOL: AN AUSTRALIAN PERSPECTIVE

Eric Williams
Charlie McKavanagh
Lecturers in Computer Education
Brisbane College of Advanced Education
Mt. Gravatt Campus, P.O. Box 82, Mt. Gravatt
Queensland, Australia

Abstract

The paper analyses the impact that the increasing use of information technology will have on the role of the secondary school. Our argument is that the computer is likely to have only a marginal effect on the educative aspects of secondary school. However it is likely to have a substantive impact on the socialization, accreditation and vocational aspects of schooling. The social, economic and political reasons for this thesis are carefully argued.

The aim of this paper is to delineate the impact of information technology on the role of the secondary education in Australia. We believe that many arguments in our paper will be applicable to the situation in the United States.

A major argument of this paper is that information technology is likely to have a marginal effect on the educative role of secondary school. This is not to deny benefits that the computer can provide as a teaching/learning tool but, because of a variety of factors, our belief is that its implementation antherefore its impact, will be limited. In contrast, due to external political and social pressure, computers are likely to have a substantive influence on other roles of the secondary school including that of "sorting" students for accreditation purposes, socialization and custodial care.

A further argument is that as youth unemployment increases, partly due to the increasing use of information technology, secondary education is likely to become more vocationally orientated. Our belief is that this is misguided and that the secondary school should in fact broaden its educative role to ensure all students are given a liberal education.

In this paper we will outline what we accept as the role of the secondary school and then analyse the impact of information technology on it.

ROLE OF THE SECONDARY SCHOOL

While rejecting many of arguments of writers such as Ivan Illich we support his view that significant aspects of the role of the secondary school are centered on socialization, custodial care and the "sorting" of students for accreditation purposes.

IMPACT OF TECHNOLOGY ON THE EDUCATIVE ROLE

The Optimist's view

Optimists such as Stonier and Conlin, O'Shea and Shelf and Dede believe that computer technology as a teaching/learning tool is unique. It is claimed that the technology is motivating because it is interactive, self pacing and provides for individualised learning. Further, through the use of graphics, sound and colour, computers can present information in an interesting and novel manner, and, unlike the teacr, computers can be programmed to be very "patient".

Further the computer will enable students to access and manipulate massive amounts of information. This will result in a major change in the relationship between teacher and student with the latter becoming more independent and autonomous.

Dede for example confidently predicts that during the next decade schools will gradually shift from a teaching/learning model based almost totally on human instruction to a new approach that combines teachers and machines.

Dede and Stonier and Conlin believe that giving students the skills and knowledge to access the vast amount of information stored on data bases means that the teacher is now, in a sense, redundant. Their notion appears to be based on an idealistic view of the student as self motivated, innately curious and able to work effectively in an independent mode. (Many teachers would question this view in the light of their own classroom experiences).

Stonier and Conlin go as far as to argue that education will increasingly occur at home with the use of computer/information technology, the secondary school's role in education will, as we know it, disappear.

A Realistic Appraisal

We believe that the optimist view is naive and idealistic. Their arguments are based on extrapolation from technical possibilities and ignore the social and economic realities of the situation.

The next part of the paper will set out why we believe that computers will have a marginal effect on the secondary school's role in teaching and learning. The reasons centre on software, availability of computer resources, teacher attitudes, lack of skills and knowledge about the value of the computer in the curriculum, and the organisation of the secondary school.

In terms of the curriculum hundreds of computer programs would be necessary to cover just a small portion of the secondary syllabus. This would be an enormous cost, in these times of budgetary constraints ow likely is this to happen? Even if more programs were available there are not enough machines in schools to enable students and staff to have sustained periods of time on them. Fitzgerald et al estimate there are 35000 computers in Australian secondary schools which on average means there are only 14 computers in each high school. Considering that many of these high schools have over a thousand students the resource to pupil ratio is likely to remain a major problem for some time to come. With the life of machines in the vicinity of five years the costs become prohibitive.

In terms of teacher's attitudes to the use of computers in the curriculum it is significant that, according to the survey conducted by Fitzgerald et al, 72% of users in Australian secondary schools were concentrated in the maths/sciences departments. Few teachers from the other subject areas seem interested or willing to become involved with the technology. Further, research findings by Firkin et al in a selected sample of secondary schools in Victoria, Australia, conclude that "While not openly hostile to computers, there was a sense of powerlessness in the area of computers which was reflected in low expectations, vague statements and a sense of indifference amongst the majority of teachers in the schools studied."

The reality of many Australian classrooms is that many staff are not able to start a computer nor to use one productively to support curriculum goals. The latter assumes the teacher has had the opportunity to build a set of cumulative experiences with the computer in a classroom setting so that they can understand how and where the different modes of the computer best "fits into" the curriculum. Few teachers have had such experiences.

Many practitioners argue thamore inservice training would solve the problem. The implication being that all that is necessary is to give the teacher the skills and knowledge about the computer and they will then integrate it within their curriculum. Providing a rational argument that computers stimulate interest and motivation does not mean that teachers, acting selflessly out of a conviction of professional committment, will then enthusiastically adopt computer technology in their teaching.

Inservice education is expensive, time consuming and often problematic in terms of success. The changes needed involve values, attitudes and teaching styles not just providing skills and knowledge. Changing values and attitudes is not an easy goal to attain in the short term. Given the competing demands on a shrinking educational budget how much can realistically be achieved?

Another reason for the likely marginal impact of computer technology on the educative role of the secondary school centers on the structure and working conditions of schools in general. The way salary and conditions are determined for teachers in Australia means that there are no extra rewards for being innovative. The teacher who leaves work at five past three is paid the same as the one who stays onto five o'clock to learn how to run new software.

Like any innovation, the introduction of computers at the school level, will take time. Yet anyone who visits our schools soon realises that the emphasis is usually on the present and the immediate, with little time for reflection, experimentation, planning and evaluation which are essential for successful innovation.

Further the current structure of schools which can be characterised as bureaucratic, hierarchical and authoritarian is hardly conducive to change.

We conclude that the computer is likely to have only a marginal effect on the traditional role of the school in the teaching and learning mode. This cannot be said for the other roles of the secondary school, such as socialization, custodialare, accreditation and vocational preparation.

IMPACT ON THE SOCIALIZATION ROLE THE SECONDARY SCHOOL

Writers such as Young and Apple have alerted us to the role of schools in maintaining the inequalities in our society. Our belief is that the use of the technology is likely to reinforce such values rather than be a new means of challenging them.

Willis points out that a number of teachers stated that computer literacy courses conducted in Australian schools were "middle class" in nature because the content offered in such courses (computing in small business, banking etc) do not relate to "working class" experiences. Willis further argues that the wealthier students, who had computers at home, often influenced the school's computer education curriculum. For the poorer students the computer literacy courses "... rather than overcoming the "disadvantage" of some children, served to reinforce their felt inadequacies so that eventually they withdraw, in effect if not in fact."

Girls can be particularly disadvantaged in gaining access to computer knowledge and skills. Whether it be lack of good role models, harassment from some male students and teachers regarding control of the technology or sexism in textbooks. Gender has become an important determinant in who gains skills and knowledge in computer technology. Is America any different?

Even those courses which are more vocationally oriented such as wordprocessing in the Commercial Studies area will result in many girls being channeled into low level formation workers with few prospects for advancement or tenure.

Our conclusion is that, in terms of class and gender, the roles we assign groups in our society in relation to the technology are often reinforced rather than being questioned by the secondary school.

Another area for concern is the subtle and at times not so subtle socialization that occurs in schools relating to the dominance of the scientific, mechanistic approach to problem solving in our society. For example, in mathematics and science, problem solving is too often taught as if it is a rational, logical, individual process without any political or social value connotations. Unfortunately in the secondary school computers are generally placed in the mathematics and science area and are, according to Fitzgerald et al, most frequently used for engaging students in solving problems in programming.

The technology is being used to foster an ideology which takes a very narrow view of what are "important" problems and the processes to be used in solving them.

Our belief is that the most significant problems we face as individuals and a society are interpersonal and political in nature. Yet these are often studiously ignored by many Australian secondary schools. The controversies that might occur would be anathema to many teachers, parents and administrators.

IMPACT ON CUSTODIAL CARE ROLE OF THE SECONDARY SCHOOL

A major role for the secondary school in our society is a custodial one, it is likely that the technology will have a major impact here.

There are many instances where technical innovation has affected the employment prospects of our youth. For example the computerization of accounting, clerical procedures, autobanking facilities and communications have led to job losses for junior office workers, technicians and telephone operators. The same process is occurring in many retail businesses.

Given these factors, the traditional role secondary schools have in the area of custodial ca of students is likely to be extended because of the decreasing number of jobs available to teenagers. The pressure for this is likely to be more political and economic than educational; our current Australian Federal Government understands only too well the political costs of high unemployment among our youth. Encouraging students to stay on in school is likely to be politically and socially advantageous. Over the last five years the number of students that have been staying on at secondary school has increased quite markedly. Our contention would be that the effects of unemployment have had an effect.

IMPACT ON THE VOCATIONAL ROLE OF THE SECONDARY SCHOOL

Schools have been attacked by politicians, employers, parents and students because the curriculum does not emphasise vocational outcomes. What is required, so the argument goes, is to provide students with the skills and knowledge that will enable them to quickly gain employment. Some politicians particularly have found it convenient to blame schools for the high level of youth unemployment and not the mismanagement of the economy.

This argument is complemented by another which states that for Australia to maintain its standard of living there must be a restructuring of the economy to take advantage of the many benefits of information technology. To do this successfully "conventional wisdom" states we must have a highly trained and retrained workforce. We question that wisdom.

Sweet points out that, in Australia the "highest rates of growth have generally occurred in work which, by virtue of its frequent part time, casual, irregular and deskilled nature, makes least demands on youth capacities". By this he is referring to such jobs as fast food outlets, cleaning and serving on tables. Further "It should be notedhat employment loss has been heavy in many occupations requiring extended formal preparation, whereas employment gains have been greatest in occupations requiring minimal vocational preparation."

Where there has been an increase in jobs that require some skills the push for this has been more to do with government election promises than technological demand. Evidence of this is Sweet's statement that in the 1970's the largest increases in employment in Australia were for public servants -- librarians, nurses, teachers and social workers and not 'high tech' workers.

Our secondary schools are now under increasing pressure to provide the vocational skills supposedly necessary to gain employment. The danger is that the pressures on the schools to produce students with marketable skills ends up doing nothing of the kind. Skills that can be taught rapidly become obsolescent with changes in the technology or are irrelevant to the lowly paid, casual and non-unionised jobs which young people are forced to take.

Further evidence suggests that our economy will not need a highly trained workforce. As Cole states, the Australian workforce requires relatively few highly skilled people, evidence of this is supplied by the Australian Department of Employment and Youth Affairs which states that less than 26% of jobs in the workforce require more than 6 months formal or on the job training.

Some argue that this will change as our economy becomes technologically more sophisticated. The American experience can be cited to reject this view. Grubb points out that the Bureau of Labour Statistics in the USA show that the demand projections for high tech jobs in 1982-95 are quite small. Tre will be a shortfall of approximately 200,000 electrical/electronic engineers, system analysts, out of a workforce of 119 million. What this implies is a change in engineering schools in higher education -- not a wholesale revision of the secondary school system.

IMPACT ON THE ACCREDITATION ROLE OF THE SECONDARY SCHOOL

Our belief is that the secondary school should not be used as a means to "sort" students socially and politically based on an accreditation process. Particularly when it is claimed to be necessary on the grounds of increasing demand for more "highly qualified" workers due technological developments.

The issue of credentialism versus education is taken up by Grubb who argues that knowledge and skills are increasingly valued by parents, students and many teachers in a strictly utilitarian approach. That is, the knowledge and skills are not valued for their own sake but only as a means to attaining a credential which can be "traded" on the labour market for a highly paid and preferably secure job. Or provide access to a tertiary institute which enables the student to raise the value of the credential in terms of future employment prospects.

This often leads to a rejection of the so called "soft option" subjects in the humanities because it is the "hard options" such as mathematics and science which supposedly "get the jobs".

The problem here is that we are in danger of producing a generation of narrowly trained school leavers who have no understanding of, or means of gaining an understanding of the problems we confront as a society, whether they have to do with technology and unemployment or other political and social issues.

Increasingly the whole process in secondary school becomes orientated to training, not to education in its broadest, liberal sense. History, social studies, literature, music and art have no supposed "economic value" therefore the danger is that they will have reduced status and resources. Yet we believe that these arehe very subjects which need to emphasised in a society increasingly dominated by economic determinism, technology and scientism.

What is even worse is that this reemphasis of the secondary school's role is simply wrong. The majority of the workforce will not need mathematical abilities, advanced scientific or computing programming skills. The jobs that show any real growth are in the low tech, poorly paid, non unionised, "dead end" jobs such as fast foods and shelf filling.

The situation is unlikely to get better. Squires points out that educational attainment has far outstripped the technical requirements of various occupations. What we have is a fundamental confusion between increasing social rather than economic pressure for credentialism and hence the supposed need for increasing education which increasingly is seen to exclude the humanities.

For those who might argue that increased schooling means that the student is more capable at an intellectual level and therefore a more useful worker, the evidence suggests otherwise. Squires makes some very telling criticisms, backed by admittedly anecdotal evidence, which suggest that, while employers have been demanding higher qualifications, there is little evidence to indicate that those more highly qualified in business do better than those who are not. Further evidence is cited to show that the vast majority of workers learn the skills and knowledge necessary for their occupation "on the job".

Surveys show that employers are not overly impressed with the cognitive aspects of a qualification. "Invariably such non-cognitive traits as personality, motivation, ability to get along and work with others are rated higher than indicators of cognitive ability like previous work experience, knowledge of a specific subject, or grades". This does not mean that employers are uninterested in the technical expertise of the potential employees but that, in terms of priority, it is the behavioral side that is most important, more schooling implies greater stability and a more easily controlled worker.

CONCLUSION

In conclusion we believe that information technology is only likely to have a marginal effect on the educational aspect of secondary schooling. As regards custodial care, accreditation and socialization the impact of the technology is likely to be much greater but, from our perspective, it will be an adverse influence.

REFERENCES

Apple, M. *Knowledge and Control*. Routledge and Kegan Paul, London 1979

Cole, P. "Schooling and Labour Market Data: ACurriculum Response Form In-Service Program." Paper, Victorian Institute of Secondary Education 1981

Dede, C. "The Likely Evolution of Computer Use in Schools" in *Computer Literacy for Teachers*, Tashner, J. (ed.), Oryx Press 1894

Firkin, J., Davidson, M., & Johnson, L. "Computer Culture in the Classroom - Case Studies of Ten Metropolitan and Rural Schools." Victorian Instiitute of Secondary Education, Melbourne 1985, p. 12

Fitzgerald, D. et al. op. cit. p.19

Fitzgerald, D. op. cit.

Grubb, W. "Responding to the Constancy of Change - New Technologies and Future Demands on U.S. Education." National Institute of Education, Stanford University 1985

Grubb, W. op. cit.

Illich, I. *Deschooling Society*. Penguin 1971

O'Shea, T. & Shelf, J. *Learning and Teaching with Computers - Artificial Intelligence in Education.* Harvester Pres 1983 p. 60

Reimann, J. "What About Girls?" paper presented at the Audstralian Comuter Education Conference, 1986, Melbourne

Squires, G. "Education, Jobs, and Equality: Functional and Conflict Models odf Social Stratification in the United States." *Social Problems* April 1977

Squires, G. op. cit. p. 3

Squires, G. op. cit. p. 11

Stonier, T. & Conlin, C. *The Three C's: Children, Computers, and Communication* John Wiley & Sons, Chichester 1985 p. 23-24

Stonier, T. & Conlin, C. op. cit.

Sweet, R. "Australian Trends in Job Skill Requirements: A Critique of the Current Orthodoxy." U.S. - Australian Joint Seminar The Future Impact of Technology on Work and Education, Monash 1984, p. 13 & 15

Willis, S. "Computers in Education: Equity and Access" Paper prepared for the Australian Association for Research in Education Annual Conference 1986, p. 6

Young, M. *Knowledge and Control - New Directions for Sociology of Education.* Collier MacMillan London 1971

CONSEQUENCES OF DECLINING ADMINISTRATIVE SUPPORT FOR ELEMENTARY SCHOOL INSTRUCTIONAL COMPUTING

John F. Beaver
Buffalo State College
1300 Elmwood Avenue
Buffalo, NY 14222

The late1980s have produced increasing and competing demands for the existing elementary school funds. During these years, many schools have had fewer resources available to them. Clearly, in these "belt tightening" situations, some programs will receive insufficient support. How have instructional computing programs fared in this environment of financial instability?

Although educational computing received strong public support in the recent past, an increased emphasis on drug education, teen pregnancy, AIDS awareness, and other current social issues may have caused elementary school administrators to shift their funds from instructional computing to other programs. How prevalent among administrators is the feeling that they've already purchased the necessary hardware and now other important educational issues should take precedance. Furthermore, are extant microcomputers being well used or has a lack of continued support caused them to be relegated to the back shelves with other technological relics of the past?

As school administrators face increased calls for budgetary accountability, they must also justify computer use in ways that are educationally sound and fiscally supportable. How can computers be used in ways that justify the expense they represent? These questions and related issues suggested the need for a study to determine how leaders of high quality elementary school instructional computing programs use their machines instructionally to ensure that they get the maximum "bang" for their computer bucks.

During the late fall of 1987, a study investigated the instructional computing practices of elementary schools selected for the high quality of their programs. The study included an examination of the schools' budgetary practices pertaining to their instructional computing programs. The manner in which school leaders allocate their available funds speaks volumes about the importance they afford to each budgetary item. The opinions of leaders whose elementary school instructional computing programs are considered high quality can be especially useful to any school administrator seeking advice on how best to allocate the limited funds available for instructional computing.

The national search identified 73 elementary schools with high quality instructional computing programs. The Office of Technical Assessment (OTA) in Washington, D.C., supplied a list of districts from around the country known for their support of instructional technology. Representatives from these districts were contacted and asked to identify the elementary schools in their areas with the most outstanding instructional computing programs. These recommended schools were therefore considered the *creme de la creme* in terms of their instructional computing programs.

After selection, these schools were then surveyed (with a 70% response rate), using an instrument designed to gather data on selected topics relating to instructional computing. Among the elements investigated were characteristics relating, either directly or indirectly, to budgetary practices: the presence or absence of computer expenses as permanent line items in the budget, details of the 1986 and 1987 computer budgets, and the total expense on computer related items.

The first issue considered whether computer expenses were permanent line items in the schools' budgets and, if so, for how many years. The data showed that only 29% of the schools had computer expenses as permanent line items in their budgets. Of these, 29% varied in the amount of time for which that had been true. 60% of the schools had maintained the line item for four or more years, 27% for three years, and the other 13% for only two years. However, nearly **two-thirds** of the schools surveyed (65%) indicated that computer expenses were still not permanent line items in their budgets.

The next issue involved the manner in which available funds had been allocated for 1986 and 1987 computer related expenses. The available data were manipulated to determine the total computer related costs per student for the two years considered. The resultant per student figures permit easier analysis by avoiding the complicating influence of student body size. The percentage distributions of the per student budget information for 1986 and 1987 are presented in Table I. Table II lists the percentage distribution for the average annual per student expenses.

In Table I, the two columns to the right of the per student figures illustrate the percentage of schools spending those amounts in 1986 and 1987. The figures for 1986 and 1987 show the total per student computer expenses reported by respondents, including staff development and equipment maintenance costs. In Table II, the figures represent the costs incurred annually during the development of the instructional computing programs (total cost divided by the number of years computing; that answer divided by the total number of students). The average annual per student figure includes only the costs of hardware, software and computer related materials. Both tables reveal steadily declining expenditures, first from the average annual expenses to the 1986 expenses and, still further, to the 1987 expenses.

The itemized budget data for the years 1986 and 1987 are presented, using means and medians, in Table III. The table indicates, in an approximate fashion, the manner in which the available funds were allocated by school leaders for the two years. That table also shows a steady downward trend in the per student expenses for the last several years, particularly if the average annual per student expenses data are included in the comparison. For instance, in 1987, 64% of the schools supplying valid data spent less than $10 per student on computer related items. In 1986, 52% of the schools spent less than $10 per student. Only 12% of the schools spent less than $10 per student during the years in which they were developing instructional computing programs. These dramatic declines suggest a clear retreat from the financial support levels that existed in the recent past.

The conclusion that expenses have declined in recent years is graphically illustrated by an examination of the means and medians for the percentage distributions presented in Tables I and II (see Table III). The mean for the average annual expenses was $28.90; the mean for 1986 expenses was $18.75; the mean for 1987 was $15.02. The pattern is even more striking for the medians since the presence of several large outliers is less significant. The median for the average annual expenses was $20.00; for 1986, it was $8.80; for 1987, it dipped to $4.90. Thus, after five years of roughly $20 per student annual expenses to develop the program, the typical school expense dropped to less than $5 per student by 1987.

The importance of this dramatic decline may be mitigated, to an extent, due to differences in the wording of the three relevant survey questions. In determining the average annual expenses, the data gathered reflected expenses--regardless of the source of the funds. The data used to determine the 1986 and 1987 computer budget figures, however, reflected only the monies specifically available for allocation by the building principal. Therefore, the 1986 and 1987 figures do not reflect funding sources outside the building principal's influence. Nevertheless, it is worth noting that the average annual expense figures do not include either staff development or equipment maintenance costs, whereas the figures for 1986 and 1987 do. Thus, the decline in per student expenses is evident despite the inclusion of additional budget categories in the 1986 and 1987 figures.

Most likely some portion of the total funds used to develop the high quality instructional computing programs came from sources other than the principals' budgets--for instance, fund-raisers or grants. However, if the large differences detected were mainly attributable to outside funding sources, then those sources must have accounted for the majority of the available funds over the years that the programs were being developed. When the 1986 and 1987 figures are compared, the declines of 56% between the two medians and of 25% between the two means (from 1986 to 1987) suggest that the budgets may fluctuate widely. Unfortunately, the study lacked the data necessary to clarify questions of budget allocations beyond the two years specified.

The final issue involved the manner in which the monies available for computer expenses were allocated for 1986 and 1987. Again, the data were examined using per student figures to enhance their value for further interpretation. More than one-half of the schools spent less than $10 per student for hardware in both 1986 (51%) and 1987 (53%). Well over half of the schools spent lower than $2 per student for software in both 1986 (68%) and 1987 (57%). Even less was expended on staff development, with the large majority of schools spending less than $1 per student in 1986 (61%) and 1987 (67%). Finally, equipment maintenance expenses were the smallest part of the budgets, with the majority of schools spending no money at all on maintenance either in 1986 (59%) or 1987 (51%).

The major budget item both years was the purchase of hardware. Expenses for hardware were followed sequentially in importance by expenditures for software, staff development, and equipment maintenance costs. The sizeable differences, proportionally, between the means and medians for all the budget items reflect the influence of very large outliers which had a much greater mathematical impact on the means than on the medians.

Table IV displays the 1986 and 1987 means from Table III to examine the proportional allocation of the available funds. This table, makes clear that hardware expenses accounted for a large majority of the computer budget expenses in both 1986 (65%) and 1987 (73%). Furthermore, while software costs have remained the second largest expense, the portion of the total budget spent on software dropped from 24% in the 1986 to 15% in 1987. Staff development and equipment expenses, although accounting for a minimal part of the budgets, both remained relatively constant over the two year period considered.

The average annual per student expense figure shows a strongly positive correlation with the average per student expenses for 1986 and 1987 (the amounts for those two years averaged). The schools supplying the data necessary for calculations produced a Pearson Product-Moment Coefficient of Correlation (r) of .77. This finding supports the supposition that the pattern of declining computer expenses is highly consistent across the entire sample of schools. The relationship is illustrated with the scattergram depicted in Figure I.

The study demonstrated that computer expenses are not yet permanent line items in the budgets of more than 70% of the schools studied. The median amount that school leaders spent each year for computer related expenses was roughly $20 per student during the years that they developed their computing programs, but the per student figure dropped first, in 1986, to $8.80 and then further, in 1987, to $4.90 per student. Of the $4.90 per student, budgetary allocation was divided between $3.57 for hardware, $0.74 for software, $0.44

for staff development and $0.15 for equipment maintenance.

The per student budgetary amounts supplied by the schools showed a noticeable difference from the figures supplied by either Moursund and Ricketts (1987) or Levin (1986). Moursund's and Ricketts' study suggested that per student expenses would total $8 for hardware and software alone (with expenditures tripling in a five year period). Levin's findings that hardware costs accounted for 11% of the total computer expenses differed substantially from the percentage discovered in this study--that is, nearly 75% of the budgets.

Levin's research indicated that schools spent between $119 and $431 per student on software, a figure that is significantly higher than any amount which could be projected from the findings in this study. Additionally, Levin reported that providing students with 10 minutes of computer assisted instruction per student per day cost roughly $120 a year, while Hawley et al (1986) found costs that were approximately one-third of those Levin reported when the instructional setting examined was a classroom instead of a lab. But even Hawley's figures were substantially higher than those found in this study.

The per student expenditures are especially noteworthy in light of this study's other findings which indicate that school leaders are considering a shift in program emphasis from computer assisted instruction to a greater use of computer applications. Application programs are frequently more hardware-intensive than computer assisted leaning programs. Instructional computing programs will need more hardware and, concurrently, more--not less--financial support. Each school's instructional leaders must supply the necessary budgetary support. Obviously, the required support will not be provided by schools with continually declining computer budgets, nor by those lacking computer related expenses as line items in their budgets. Neither situation can provide the on-going financial support necessary for instructional computing programs to succeed.

Educational leaders must establish long range plans to develop their instructional computing programs, take into account the unstable nature of computer budgets, and attempt, where possible, to eliminate the factors which create budgetary instability. In particular, leaders must strive to include computer expenses as permanent line items in their budgets. School leaders should also strive to stabilize the expenditure levels of computer budgets and raise them to an extent which permits implementation of the programs they plan to support. In many cases, this will require substantial increases in the existent budgets. In particular, funds should be provided for comprehensive inservice programs designed to encourage the integration of computers into the existing curriculum by regular classroom teachers.

The budgetary restrictions of the past several years have apparently had a great impact on the level of computer expenditure in the schools sampled. If this study's findings are generally indicative of budgetary practices in elementary schools, then computer budgets are now falling quickly instead of rising rapidly. School leaders must strive to alter this pattern with every means at their disposal. They must attempt to engage parent groups, business and industry leaders, and national, state, and local legislators in efforts to generate increased financial support. Educational leaders cannot afford to view instructional computing as yet another passing phase for which they have already paid their dues.

BIBLIOGRAPHY

Hawley, David E., J. D. Fletcher, and P. K. Piele. "Costs, Effects and Utility of Microcomputer-Assisted Instruction." Eugene: Center for Advanced Technology in Education. November, 1986.

Levin, H. M. "Cost-effectiveness of computer-assisted instruction: Some insights." Stanford: Stanford Education Policy Institute Report. September, 1986.

Market Data Retrieval. *Microcomputers in Schools, 1983-84: A Comprehensive Survey and Analysis.* Westport: Market Data Retrieval. 1984.

Moursund, Dave, and Ricketts, Dick. *Long-Range Planning For Computers in Schools.* Eugene: Information Age Education Press. 1987.

Taylor, Robert, Ed. *The Computer in the School: Tool, Tutor, Tutee.* New York: Teachers College Press. 1980.

Table I: Total Per Student Computer Expenses In 1986 and 1987

Total Expenses per Student for Computer Related Items	% of the Total Schools in 1986*	% of the Total Schools in 1987*
Less than $5	27%	43%
Between $5 and $10	20%	10%
Between $10 and $15	14%	8%
Between $15 and $20	6%	4%
Between $20 and $25	4%	0%
Between $25 and $30	6%	4%
Greater than $30	14%	14%
Total % of Schools	91 %	73%

* Some data was not provided

Table II: Total Per Student Computer Expenses--Annual Average

Total Expenses per Student for Computer Related Items	Total Percentage of the Schools: Annual Average*
Less than $5	6%
Between $5 and $10	6%
Between $10 and $15	12%
Between $15 and $20	14%
Between $25 and $30	2%
Greater than $30	32%
Total % of Schools	72%

* Some data was not provided

Table III: Itemized Per Student Computer Expenses

Specific Item From the Computer Budget	Mean 1986	Mean 1987	Median 1986	Median 1987
Hardware	$12.20	$10.90	$7.61	$3.12
Software	4.51	2.24	1.34	1.20
Staff Development	1.64	1.36	0.05	0.00
Maintenance	0.40	0.52	0.00	0.00
TOTAL BUDGET**	18.76*	15.02	8.80	4.90

* Due to Rounding

** Data are Total Budget figures, not column summations

Table IV: Percentage Allocation of Computer Funds

Specific item from the Computer Budget	Percentage of the Total 1986 Budget	Percentage of the Total 1987 Budget
Hardware Purchase	65%	73%
Software Purchase	24%	15%
Staff Development	9%	9%
Maintenance	2%	3%
TOTAL BUDGET	100%	100%

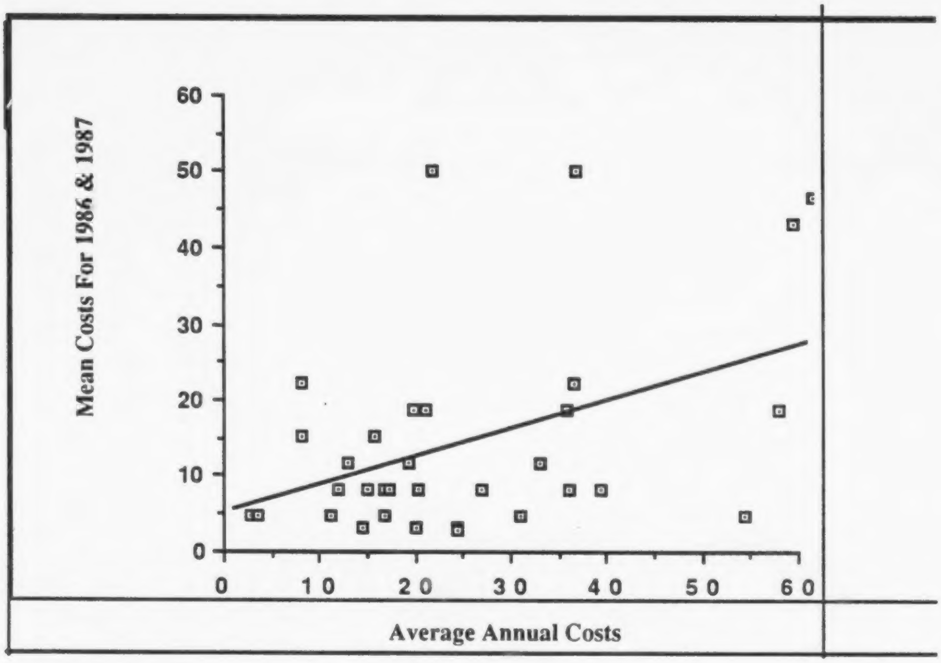


FIGURE I: BUDGETARY CORRELATION--Annual Average vs Recent Years

INFORMAL COMPUTER LEARNING IN MUSEUMS

Adeline Naiman, Chair
Director of Education
The Computer Museum
Boston, MA

Inabeth Miller
Director of Outreach
Boston Science Museum
Boston, MA

ABSTRACT

The nature of schooling has changed dramatically in recent decades. Continuing education constitutes a large part of higher education as adults return to college to acquire skills they need to improve their lives. Enrichment of the K-12 curriculum now includes the use of resources beyond the classroom. The development of low-cost mass communications and instructional devices has broadened the array of enrichment approaches teachers may draw on to reinforce conventional methods. More important, "informal" learning has come to reinforce (or in some cases replace) formal instruction.

Nowhere is this expansion of learning options more visible than in the changing nature of museums. From their original role as repositories for static artifacts, museums have become active sponsors of the learning process. Presentations, docents, exhibit audiotapes, and special programs for schoolchildren, teachers, and the general public are only part of the new responsibilities museums have taken on. Moreover, schools, parents, and other visitors count on the museum to fill this function in their lives. Carefully planned school visits are a staple of every museum's offerings and go well beyond a simple field trip.

In the world of educational computing, the museum has perforce had to fill voids in school curricula, teacher training, and public understanding of technology. Many science museums and children's museums now include exhibits and instructional programs on computers and the many computer-centered technologies. The Computer Museum, the only museum in the world devoted solely to the history, technology, and uses of computers, serves thousands of schoolchildren and their teachers in a variety of ways. The Science Museum in Boston has its own Computerplace, where hardware and software may be explored by the public.

In settings like these, interactive exhibits are the means through which even very young children (as well as their parents) can begin to understand and deal with some of the major technical forces that are reshaping our lives. The initial timidity many adults feel towards computers is quickly overcome once they have their hands on a keyboard or mouse and find they can control what happens.

The critcal element is "locus of control". The museum setting is different from the claasroom. It is carefully designed to support learning, but the atmosphere is not threatening and judgemental. Nobody is watching or giving grades. It is okay (and not embarrassing) to try something again and again, to make mistakes and learn from them. The visitor does not have to depend on autority in order to take charge, though help is available. There are no time constraints. The nature of computer use is that the user can set an individual pace and make independent decisions. Much formal schooling is not so supportive of a range of learning styles and preferences.

Finally, the computer is of itself a motivator of learning. Most owners of personal computers are in large part self (or peer) taught. The satisfaction of independent mastery is hard to equal. Children who resist conventional instructional media and methods often have to be dragged away from a computer they control in a museum. So do their teachers, their parents, their grandparents! "Hands on" is only part of the reason. So is the direct feedback. Best of all is the sheer pleasure of learning for oneself through the magic of the computer.

Panelists will report on exhibits, programs, and plans for using computers to further informal learning in their museums.

TEACHERS BECOMING RESEARCHERS: AN EXPLORATION IN DYNAMIC COLLABORATION

Bridget Somekh
PALM Project
Centre for Appplied Research in Education
University of East Anglia
Great Britain

ABSTRACT

The teacher-as-researcher movement in Britain began with the work of Lawrence Stenhouse, who believed that teachers' involvement in research was a means of overcoming the resistance to curriculum change. Later John Elliott, who had worked with Stenhouse on his Humanities Curriculum Project, carried out seminal work on the Ford Teaching Project which involved teachers using action research to identify methods of teaching to promote Enquiry/Discovery learning. Several other teacher action research projects followed, notably the Teacher-pupil Interaction and the Quality of Learning Project (TIQL) in which teachers researched methods of teaching for understanding, particularly in the context of assessment.

The Pupil Autonomy in Learning with Microcomputers Project (PALM) sets out to build on this earlier work. It is focussing particularly on the nature of student learning when microcomputers are used as a tool. It addresses questions about the quality of student learning as opposed to student activity; the role of the teacher when a micro makes a third partner in a teaching-learning triangle; the potential for micros to encourage student autonomous group work (and the role of the teacher in facilitating this kind of learning). In broad terms it is looking for evidence that microcomputers are enhancing the learning environment within British schools, now, with present levels of resourcing and teacher training.

PALM is involving teachers in 23 schools in carrying out Action Research in order to look at these questions. Their work is facilitated by a central team based at the University of East Anglia, comprising myself and three teachers, Jon Pratt, Erica Brown, and Bob Davison, who are seconded to PALM for two years. This central team works closely with John Elliott, the Project Director and with the Inspectors for Information Technology in the three participating Local Education Authorities, as well as receiving support from representatives of the Microelectronic Education Support Unit (MESU), a unit within the National Council for Educational Technology (NCET), which is funding PALM.

The paper will address major issues in working with teachers as reserchers within the PALM Project. In particular, it will focus on the following:

1. The process involved in establishing a collaborative research team within a school: negotiating the involvement of teachers, the role of a teacher coordinator within each school, the establishment of research partnerships, the relationship of the project to the formal and informal structures of the school, the organization of meetings.

2. The need to establish teachers in control of their own research: issues of confidentiality of research data within the political context of the school, the local education authority, the project and NCET (the funding body).

3. The relationship between teacher research, curriculum development in Information Technology and In-service training of teachers. In particular, the constraints placed upon teacher researchers in the initial stages by their inexperience in using microcomputers.

4. Barriers to computers as an innovation which have arisen within the initial experience of the project, and ways in which involvement in the action research process have helped to overcome these barriers.

5. Issues arising from early identification of research questions. For example, the extent to which these are identified by the teacher researchers themselves and the extent to which they are identified within and across schools in order to give the work of the project as a whole coherence.

6. Issues concerning the quality of the research work being undertaken, keeping the focus of action research as a methodology leading to changes in teaching and learning styles, rather than a set of techniques and procedures. Arising from this there will be some discussion of the development of

materials designed to enable teachers to acquire action research skills quickly and easily.

7. The relationship between the central team and individual teacher researchers: the process of facilitating teacher action research. A variety of approaches are being used, depending on the nature of the school, the individual teachers involved, and the needs of the three teachers on the project team to develop skills in both research and facilitation.

8. Issues relating to the need to build in structures for early dissemination of research findings. In particular, the need to prevent research teams in schools from becoming isolated from colleagues; the need to develop confidence in teachers presenting their work in either written or oral form; and the need to keep the members of the Local Education Authority support services and inspectorates fully informed in order to enable dissemination at a later stage to other schools.

The paper will end by referring briefly to some examples of early research work carried out by teachers. This will lead into consideration of important issues relating to teachers reporting their own research, both through PALM publications and through the Bulletins and Conferences of the Classroom Action Research Network (CARN).

THE IMPORTANCE OF TEACHERS IN CENTER FOR TECHNOLOGY IN EDUCATION RESEARCH

Karen Sheingold, Chair
Center for Children and Technology/
Center for Technology in Education
Bank Street College of Education
610 West 112th Street
New York, NY 10025

ABSTRACT

In its research and development, whose overall goal is understanding how to integrate technology to improve students' learning, the CTE is learning from and working with teachers. This symposium will describe four ways in which teachers will contribute to the Center's work over the next five years. First, studies will investigate what factors teachers believe contribute to the successful integration of technology, and what constitutes the knowledge that teachers have who are successful at integrating technology. Second, CTE staff are codesigning preservice courses that integrate technology with subject-matter curricula. Third, collaborative school-based experiments on technology integration are being codesigned with teachers in selected sites. Fourth, the CTE staff is investigating how school restructuring can be related to the effective integration of technology. Specifically, through its school-based research, it is exploring the school-wide conditions that support teachers' thoughtful planning and using technology for instruction.

PANELISTS

Jan Hawkins
Cornelia Bruner
Margaret Honey
Center for Children and Technology/
Center for Technology in Education
Bank Street College of Education
New York, NY

DEVELOPMENT OF 2+2 PROGRAMS IN COMPUTER INFORMATION SYSTEMS: AN EXAMPLE OF SECONDARY AND COMMUNITY COLLEGE PARTNERSHIP IN CURRICULUM ARTICULATION

Ann Roskow, Chair
Anita Defina
Ulster County Community College
Stone Ridge, NY

ABSTRACT

A 2+2 Partnership Program has been established at Ulster County Community College (UCCC) in order to articulate with area high schools and develop prerequisite curricula for the technology career programs at Ulster. As outlined in Dale Parnell's book, *The Neglected Majority*, the 2+2 Project acts as a tech/prep alternative to traditional college/prep. It targets high school students from the two middle quartiles of the population, many of whom have potential to succeed but often lack direction. The Computer Information Systems programs (CIS) at UCCC are designed to prepare students for entry level positions in a variety of computer related areas. The components of the program also prepare students for transfer to selected four-year schools.

The presentation will cover:

- An explanation of the 2+2 concept
- A brief historical overview of the development of 2+2 programs at UCCC
- The development of articulation in the CIS programs at UCCC with the high schools
- A review of the progress of the program
- A sharing of materials and resources

SPONSOR: ASCUE

THE IMPACT OF COMPUTERS ON CLASSROOMS*

Margaret Riel
Director of Research and Development
InterLearn

Abstract

Do computers change education? This is the underlying question of the work that is reported in this paper. Data collected from four classrooms is used to assess the impact of computers on (1) classroom organization, (2) interactive patterns between teachers and students and (3) student learning. Computers, by themselves, do not lead to the restructuring of education but computer technology in the hand of good teachers can be an extremely effective tools in the education process.

Computers in Classrooms

Caught in the midst of what some have called a computer revolution, the question remains: What is the impact of computers on classrooms? Those who dismiss the notion of a computer revolution assert that classroom culture will dictate the way in which computers are used with little or no change to educational practice (Noble 1985, Brophy & Hannon, 1984, Tyack & Hansot 1985). Those who see technology as the newest tool for the wholesale restructuring of education, invoke visions of the modern classroom that computers will soon create (Papert 1980). Our study of computer-using classrooms leads us to dismiss both interpretations as overstated; we are more inclined to adopt the view that characterizes the relationship between classroom organization and computer use as a mutually influential one.

To understand the changes that take place when a computer is added to the educational environment in classrooms, we placed a single computer into each of four classrooms and examined the changes that took place in teaching and learning as this new technology was incorporated into the educational practices over a school year (for a more detailed report see Mehan, Riel & Moll 1985).

The teachers in this project shared a belief the computers should be integrated with their instructional and curriculum plans. They did not see a computer as an isolated piece of educational technology that students need to learn to master through programming. Nor did they see it as a game machine to reward students who finished their regularly assigned lessons. It was their goal to find ways to have the computer function as part of the classroom environment used in much the way other educational tools, tape recorders, paper, pencils, chalk and chalkboard have come to be used.

The teachers organized tasks for the microcomputer that were coordinated with tasks that were carried out in other parts of the curriculum. For example in writing, students may work on ideas for a writing project in a whole group setting or in small groups. Computers were used to help students organize their ideas into a first draft. These drafts became the reading material of response groups. Students discussed the strengths and weaknesses of their writing in small groups, working with computer printed copies. Peer editing sessions off the computer were followed by editing with the computer. In this way, the microcomputer was made part of the process, a tool to accomplish the goals that the teachers had established for their students.

The teachers were interviewed weekly over the course of the year and classroom observers took notes in each classroom 3 days a week. Examples of specific types of lessons and computer sessions were videotaped at the rate of 1-3 times a month. All the students' computer work as well as work during videotaped lessons was collected.

We used this corpus of data to explore the impact the computer on (1) classroom organization, (2) interactive patterns between teachers and students and (3) student learning.

Classroom Organization

Before the computers arrived on the scene, two of the classroom teachers taught their students primarily in whole group lessons with the other two teachers using make extensive use of learning centers and small group instruction. We expected the computer to fit easily into the learning center arrangements and to have a significant role in restructuring the whole group format to a new arrangement.

We found no significant change in the way the teachers arranged the space and used time in their classrooms when they had a microcomputer available for instruction on a full time basis. The teachers who had used learning centers extensively in previous years used this spatial and instructional configuration when a microcomputer was made available to them. The computer became one of the learning centers that students rotated though over the course of each day. The teachers comments indicated that they found it much easier to set up a computer learning activity since the computer provided both the materials and organization of the task.

---

* The work summarized here was carried out by a research team at the University of California led by Hugh Mehan. The contributions of Marcie Boruta, Luis Moll, Ann-Marie Newcomb, Robet Rowe, Marti Tum Suden and Kim Whooley to the research process are gratefully appreciated.as are the comments on earlier drafts of this paper by James Levin, Bud Mehan and Randall Souviney. The research was supported by a grant for the National Institute for Education (#NIE 6-83-0027) which does not necessarily endorse these conclusions. Requests for copies should be sent to Margaret Riel, InterLearn, 943 San Dieguito Dr., Encinitas, CA.

The teachers who used whole group methods of instruction in previous years continued to teach their classes in this manner when they had a computer available. Teachers used the computer to demonstrate concepts to the whole class. When pairs of students worked at the computer they were excused from the class activity with the instructions to "make up" any classroom work that was missed while they were working on the computer.

The absence of changes in temporal and spatial arrangements observed when microcomputers were introduced into classrooms shows how resilient classrooms are to change. The results of this study suggest that we should not be surprised if microcomputers continue to be inserted into existing classroom arrangements without leading to wholesale changes in classroom organization (Mehan 1988).

Patterns of Interaction in the Classroom

While the introduction of a microcomputer for the purposes of instruction did not modify existing spatial and temporal arrangements in the four project classrooms, the availability of a microcomputer added a new dimension of participation to the classrooms. Teachers found they could increase the time students spent working on the computer by have students work in pairs rather than individually.

Dyadic peer interaction was the new form of classroom participation that emerged from this pattern of use. Students were given assignments for work sessions at the computer by the teacher, either verbally at a whole-group orienting session, or in writing at the computer center itself. Students worked together on the assigned activity carrying out the teacher's assignments without direct adult supervision. When the pair of children working at the computer had difficulty with computer operations, teachers encouraged them to use each other as resources, consult the written instructions around the computer, or to go to other student experts for assistance.

The teachers did not dictate a particular form of interaction to the student pairs. They were left to their own devices to sort out the manner in which the task would be completed. In that sense, the students' participation in the computer activity was voluntary, not compulsory. While they were responsible for completing their assigned session at the computer, the details of how that session would be completed was left to the students. Since the teacher did not monitor the students at the computer directly, their work was not evaluated moment-to-moment or publicly, as it so often is in regular classroom lessons.

As a consequence of this additional participation structure, students developed a different sense of social relations. The students assisted each other at the computer in ways that were productive. They often corrected each other's mistakes and cooperated in the completion of assigned tasks. Dyadic peer interaction also provided social resources which facilitated learning. In language arts activities, even when neither student began an assignment with an idea of what to do, the discussion of the problem often presented the students with the way to proceed. In the process of entering text, the student who was typing was often concerned with such local issues as the spelling of a word, while the other student concentrated on more global issues such as the construction of the essay and coherence among sentences.

Verbal interaction was a particularly important medium in these situations, because the students working together talked out loud to each other. The act of verbalizing material led to cognitive restructuring on the part of the students who were attempting to explain. Verbal interaction was also important because it led students to hear different points of view, which, in turn, lead to cognitive conflicts. The resolution of these conflicts required the students to examine their own understandings and to consider different viewpoints.

Learning with computers

The computer word processing systems have been touted as possessing the solution to problems in writing because they facilitate the production of manuscripts. Printers are said to facilitate the writing process because students find the immediate production of neat, professional looking copy to be highly rewarding and motivating.

However impressive the utility of word processors and printers, these programs do not, in and of themselves, lead to improved writing. Computers alone can not solve the problem of teaching students to read and can not transform unskilled writers into skilled ones. But computers can help organize a learning environment that makes a new social organization for writing possible.

The microcomputer works effectively in language arts when tasks that are organized for it are coordinated with tasks that are carried out in other parts of the curriculum. It is the creation of functional learning environments which utilize the computer as a tool to meet educational goals, and not the computer treated as a teaching machine that dispenses knowledge to students, which has positive effects on the writing process (Riel 1985, Riel, Levin & Miller-Souviney 1986, Cohen & Riel 1988).

The availability of a computer facilitated a new social organization for reading and writing. It is at this juncture that computer technology moved beyond its role of providing a new, albeit dynamic, means to reach previously established goals, to providing a medium through which new and previously unattainable educational goals could be reached.

The teachers organized their reading, writing and grammar instructions around a newspaper-writing activity and newswire service supported by the computer (Levin, Riel, Boruta & Rowe 1984, Riel 1985). The teachers introduced a student newswire service known as the "Computer Chronicles" in their classrooms. Students from the four project classrooms joined with students in other locations to exchange their written work on the newswire. Each classroom published their local editions of the Computer Chronicles Newspaper carrying stories from their distant reporters.

The Computer Chronicles project helped the teachers establish learning environments which were organized for communicative purposes and not just as an exercise for teachers to evaluate. The presence of an audience for writing, in the form of classmates, parents and peers in Hawaii, Mexico and Alaska, was a crucial ingredient in giving students a purpose for writing. This *writing for a purpose* and not "just writing" or even writing on the computer, subordinated students' concern for the mechanics of writing to the goal of communicating clearly. The public nature of writing provided motivation for re-writing and editing, giving students increased knowledge of word processing and control over the composing process.

Dynamic support provided by the interactive capabilities of the computer resulted in improved quality and fluency. By arranging learning environments in which computer based support was gradually removed, students gained control of writing by gradually assuming the parts of the task initially accomplished by the computer.

Students in all classrooms improved their reading and writing skills beyond grade level expectations. The most impressive gains appeared in the classroom taught by the teacher who had some prior experience working with computers. The students in her classroom gained, on the average, 3 grade levels in language mechanics and 2 grade levels in language expression on the yearly administered Comprehensive Test of Basic Skills (CTBS) (See figure 1).

Our observations and intuitions suggested that the increase in students' learning was the result of their efforts at editing the work of [other] students rather than from editing their own errors. Editing the work of others increased the students' confidence since it is often easier to locate the errors of others than one's own. Students work in evaluating the articles written by other students often provided a clearer perspective for them to view their own work.

To further explore the idea that evaluating the work of other students may be a crucial variable contributing to student learning on this project, we compared the CTBS scores of the students on the first editorial board with those of the whole class. While all students in the classroom wrote articles for the newswire, a smaller number of students volunteered to serve on an editorial board to evaluate and select articles from the newswire for their publication. These students' gains were similar to the class with one dramatic exception. They gained almost 4 grade levels in language mechanics (Figure 2).

We have been questioned by disbelieving teachers who feel that it is impossible for students to learn four years worth of language mechanics in a single year. Our response is to change the focus. We think that it takes four years to teach one year of language conventions when these skills are taught out of the context of purposeful writing.

The changes that we saw in learning in this classroom cannot be causally linked to a computer. They were the effects of the creation of a functional learning environment by a skilled teacher who had some experience in working with computers. Students did not improve as dramatically in classrooms where some of these features were absent (e. g., where there was a novice computer user, or the teacher was inexperienced teaching language arts as an integrated activity). Therefore, while it is difficult to say how much of students' improvement, if any, can be attributed to the computer alone, the results of our research suggest that a combination of features (computer knowledge, teacher experience, the integration of the computer into functional learning environments) had a positive effect on students' learning.

Computers turn practice into performance

In this study teachers used a computer to create a functional learning environment for writing. The students did not just practice writing. They used the skill of writing to communicate their ideas with others who were separated from them in time and space.

Classrooms have become places were kids practice skills that will someday serve them in the adult world. Students are routinely asked to practice a skill over and over again, often with little understanding of how it will fit into their overall pattern of skill development. Even successful students characterize school, and the work done there, as boring. Teachers who construct these sequences of drills ask students for their blind trust; students frequently respond with bored attention. The result is often decontextualized learning which does not transfer to settings outside of school.

Dewey worried about this problem at the turn of the century. Formal methods of instruction, he claimed, were causing an undesirable split between what is gained from experiences with objects and people, and what is learned in schools (Dewey 1916). His solution to this problem was to find ways to help children work with the tools of the trades in educational settings in ways that modelled the activity of adults. A printing press, a weaving loom, construction tools, and farming equipment could help students acquire academic skills in the context of their use. He was not advocating vocational training, but instead placing skill development within educational experiences that make clear the need for conventions, standards or levels of precision. Students learn from the consequence of their errors. His solutions was unworkable with the technology of the past. Machinery was expensive, space consuming, dangerous and fragile.

The turn of the next century is giving new life to Dewey's ideas. Computer technology provides a tool of the trades that can be successfully imported into the classroom. A computer is relatively inexpensive compared to a printing press or a weaving loom, flexible enough to model many different types of machines, and safe and durable enough to withstand use by young children. The communication possibilities of computers offer a new strategy for social interactions that extend beyond the classroom walls.

In this study the teachers used their computers as writing, printing and communication tools. They created a writing environment that was similar to that of professional writers. Students wrote to communicate their ideas with distant audiences--others who would read their work for its content and ideas. The use of standard writing conventions becomes clear when students had difficulty understanding the ideas of others who fail to use them. When students work together to create evaluative standards for judging the work of others, they become internalized making it easier for them to judge their own writing. The reciprocal relationship between the writer's message and the reader's interest becomes clear as students see which of their articles are published in distant locations..

Writing, editing and publishing their newspaper was far from boring, it was serious work. Students had a goal that they wanted to accomplish and the teacher was became a resource who could help them. Functional learning environments embed skill development within a task that students set for themselves (Newman 1984). Skill practice is placed in context and is driven by skill performance.

Bringing computer writing technology into the classroom helped the students to see and participate in all the steps of newspaper publishing. The project was under their control. The participation and motivation of the students, and the aid of the computer tools make the task more manageable for the teacher. The computer provided a writing technology that was interactive, as well as a means for transmitting articles to distant locations and creating their publication.

Conclusion: Teachers Teach, Machines Mediate

Computers extend rather than replace teaching done by teachers. Used in the way described in this study, computers can extend the power of students to create, analyze, compare, examine and understand. Computer facilitated environments can promote creative thinking, extend systematic inquiry and problem solving, and establish important skills for cooperative work, all skills that are vital for participation in our present and future society. Students need to learn more than a set of disconnected facts, computer can help them understand including how information is collected, stored and utilized to solve problems or create new understandings.

Educational technology makes it possible to create learning situations in which students can be engaged in activities that they find interesting and exciting for their own reasons and which accomplish the educational goals of their teachers. Teachers established functional learning environments by relating the computer activities to other educational tasks the children were doing. The goal was to link computer work with other classroom work to establish a mutually supporting context in which similar skills could be applied.

Educational technology can create new avenues for social exchange and cooperative learning. Fears that computers will result in students working in isolation removed from all forms of human interaction can be dispelled by watching students in classrooms organized to promote peer interaction. Students solve problems collaborative, often with their teachers as partners.

While it may be true that exceptional teachers can accomplish the goals associated with an integrated language arts curriculum without a computer, well-designed computer software can empower good teachers by providing them with guidelines for integrating a range of new tools with their teaching skills.

Revolution is too strong a terms for the change that is accompanying the computer into the classroom. But educational technology may provide the key to a radical recontextualization of learning. Children may find that computers enable them to participate more fully in meaningful activities. They may be able to place drill and practice in a more direct relationship with performance. The recognition that accompanies skillful performances may be just what kids are missing when they dismiss school as boring.

References

Brophy, J. & Hannon, P. (1984) The future of microcomputers in the classroom. Occasional Paper No. 76, Institute for Research on Teaching, Michigan State University.

Cohen, M. & Riel, M. (1988) The effect of Distant Audiences on Students' Writing. American Educational Research Journal (in press).

Dewey, J (1916) *Democracy and Education*. New York: The Free Press.

Levin, J. A., Riel, M., Boruta, M. & Rowe, R. (1984). Muktuk meets jaccuzi: Computer networks and elementary schools. In: Sarah Freedman (ed.), *The Acquisition of written Language*, New York: Ablex Publishing Company.

Mehan, H. Microcomputers in classrooms: Educational technology or social practice? *Anthropology and Education Quarterly* (in press).

Mehan, H., Moll, L. & Riel, M. M. (1985). *Computers in classrooms: A quasi-experiment in guided change*. (Contract No. NIE 6-83-0027). La Jolla, CA: Teacher Education Program.

Newman, D. (1984). *Functional learning environments* (Tech. Rep. No. 25). New York: Bank Street College of Education, Center for Children and Technology.

Noble, D. (1985) The underside of computer literacy. *Raritan* p. 41-64.

Papert, S. (1980) *Mindstorms*. Cambridge, Mass.:MIT press.

Riel, M. M. (1985). The computer chronicles newswire: A functional learning environment for acquiring literacy skills. *Journal of Educational Computing Research, 1*, 317-337.

Riel, M. M., Levin, J. A. & Miller-Souviney, B. (1985). *Learning with interactive media: Dynamic support for students and teachers*. (Tech Rep #7). La Jolla, CA: Interactive Technology Laboratory.

Tyack, D. & Hansot, E. (1985) Truths that never happened: Technology and the classroom. Commentary. *Education Week*, September 4.

DISTANT DIALOGS: TELECOMMUNICATIONS IN TEACHER TRAINING

R. Ann Zinck
Assistant for Academic Computing
Computer Services
San Joaquin Delta College
Stockton, CA 95207

DEAR MR X,

MY NAME IS JOHN.DO YOU LIKE SPORTS? THATS WHAT I THOUGHT. I LIKE SPORTS. MY FAVORITES ARE FOOTBALL AND BASKETBALL.

I AM A BOY,NOT A GIRL,AND ABOUT 4 FT. 7 INCHES TALL. SHORT HUH? I HAVE BROWN HAIR AND I AM VERY SMART. MY FAVORITE SUBJECTS ARE READING, P.E., AND ART AND DRAMA.

I AM GOD AT SPORTS, WHICH I ALREADY SAID, AND AT ALMOST ALL SCHOOL SUBJECTS. I AM ALSO GOOD AT BUILDING.

ONE OF MY WEAKNESSES IS DIVISION BECAUSE I HATE IT. ANOTHER WEAKNESS IS ART BECAUSE I CAN'T DRAW WORTH BEANS.

P.S.-PLEASE WRITE BACK. I'M FUNNY, HUH?

FROM: JOHN

So began a distant, yet friendly, relationship in Stockton, California between John, a "smart" fifth grader from Hazelton Elementary School and Goeff, a teacher candidate in the School of Education, University of the Pacific. John and Goeff were two of thirty-one fifth and sixth graders and seventeen university students who participated in a unique "computer pal" project during the Fall, 1987. This project was a cooperative effort between an elementary school teacher, Marcia Knudsen, and myself to explore the feasibility of using telecommunications as an alternative "field" experience in a reading methods class. At the university, students enrolled in a reading methods class were asked to volunteer for this project involving telecommunications and "computer pals" from a local elementary school. In preparation for the project, Ms. Knudsen worked with her elementary students on a theme of friendship. The friendship theme became the basis for reading and writing projects in the class. From this foundation, the outreach to the university computer pal became a logical and meaningful extension of classroom activities.

Project Goals

A number of recent articles (Tennis, J. 1985 and 1986, "Special Issue--Telecommunications." *The Computing Teacher* , 1987.) on the use of telecommunications piqued my interest. Ms Knudsen and I became interested in merging telecommunications with the writing process since telecommunications seemed an excellent vehicle for creating a meaningful and exciting writing environment. The project provided the participants with a unique setting in which to learn more about children, writing, and computers. The opportunity for the teacher candidates to begin to know a child well through his or her writing is not often possible in traditional field work settings where interactions are brief, sporadic, and usually confined to group settings. Based on articles that I had read relating the trials of neophytes in the world of telecommunications (Tennis, 1985), Ms. Knudsen and I were cautious in setting goals for the project. We decided that for the first semester the following would be reasonable:

1. To determine the feasibility and utility of using telecommunications in a teacher training program.
2. To examine the impact of the "computer pal" idea on elementary students' writing.
3. To examine what teacher candidates would learn about children and their writing.
4. To develop positive attitudes on the part of teacher candidates toward the use of computers in education.

In spite of the anticipated complexities of getting an internal modem functional on an Apple //e, telecommunications seemed "doable." The elementary school already had a phone line into the computer lab, a fortunate occurence for the project. An interested and supportive dean was able to pull the financial and influential strings necessary for installation of phone lines in the School of Education's computer lab. Four modems were purchased through a special teaching incentive grant from the university, two for each school. By the end of the Summer 1987, the hardware was ready and waiting for its initiation. We had set a deadline of the first week of October to have the project operating. It was thought that this would give the elementary students time to settle into a routine and also provide enough time for the instructors to learn the mysteries of modem based communication.

Trial by Modem

Early in September, the process of sorting out the installation and use of an internal ProModem 1200A began. After getting the modem inserted in the designated slot, the next task was to sort out the jumper pin configurations. The confusion and frustration encountered at this point unfortunately became the norm for this process. Thus it bears telling. My experience with telecommunications on a Macintosh provided a decent command of the jargon and concepts related to modems. Secondly, a veteran user of telecommunications was there to provide back-up support. The first problem stemmed from a statement in the manual in regard to the placement of a jumper and pin configuration. The directions read, "If you are going to be using communications software that calls for Super Serial Card emulation (such as ProCom-A), put the jumper on the left two pins." (p.3-2) Very straightforward, except that this particular set of pins is oriented vertically. It took two more weeks of dialing the technical "support" line to crack through the busy signal to straighten out this "misprint" and find that the "lower" two pins were actually the target. The story continued with frustrations in getting the ProCom software to work as described (it still won't and a promised software update never materialized), thus obligating us to use the firmware communications due to a limited software budget. With the help of the school district computer coordinator, we tried file transfer with the built-in commands over and over unsuccessfully, thus the transfer of computer pal letters appeared impossible for the immediate future. The final blow was discovering that only the incoming side of the on-line conversations would be saved from the buffer files. Thus we were faced with tedious "cut and pastes" once the file was converted to *ProDos* ™ and then to *Appleworks*™ in order to have a document of both sides of an on-line conversation. All of these problems popped up over a period of about six weeks which meant mid-October had arrived and the project was still on hold.

With busy teaching schedules and frustrating hardware and software problems it seemed that this project would be scrapped for a semester. Fortunately, the Computer Using Educators' (CUE) Fall Conference met in mid-October, 1987 (San Jose, California). It seemed that telecommunications was the hot topic for the conference and sessions addressing all sorts of telecommunication projects were in abundance. At one session, the electronic mail simulation, Kidmail*, was demonstrated. In one wonderful piece of public domain software the project was saved. It was obvious that Kidmail could be used to initiate contact between the children and university students. A second piece of software, FrEd Sender* gave us hope for the future. FrEd Sender, a public domain program for file transfer, would allow us to bypass both the problematic ProCom communications software and the file transfer routine with the firmware.

Connection Made

Kidmail is a well-designed program that was easy for both fifth graders and novice computer-users within the university student group to learn quickly. The fifth graders were particularly taken with having a secret code to access their mail and the responsibility of remembering the code. The elementary students were limited to the use of two computers. (The 12 Apple //e systems that had been ordered for September, 1987 were noticeably late.) Thus, with a combination of a full curriculum and slow keyboarding skills, the exchange of Kidmail letters was not to be as snappy as desired. The first disks containing the children's letters were received the week of November 2. For the teacher candidates, the excitement of finally receiving the disk based mail from their young computer pals erased the frustration of the long awaited contact.

The first letter written by the children to the teacher candidates was an introductory letter. John's letter (see above) was representative in terms of the content and conversational tone taken by the children. Most letters were quite straightforward, listing their grade, age, likes and dislikes, nature and size of their family, and expressing a real interest in the same information from the university student. One letter was more than straightforward, Brad came right to the point in his introduction:

TO WHOM IT MAY CONSERN,

MY NAME IS BOB, AND I REALLY LIKE MATH. I GET ALMOST STRAIT A's. WHAT ARE YOU LIKE? HOW OLD ARE YOU? I'M IN THE 6th GRADE AND 11 YEARS OLD. I HOPE WE CAN MEET AND TALK ABOUT OUR DIFFERENCES. THEN WE WON'T WASTE OUR TIME ARGUING BOUT WHAT TO DO.

YOUR COMPUTER PAL,
BOB

Two other interesting and somewhat surprising things were discovered in the children's introductory letters. One was their genuine interest and excitement about having a college student to as a computer pal. This was touchingly illustrated by the following excerpt from Tim's letter:

DEAR FRIEND,

HI, MY NAME IS TIM. AND I'VE BEEN WANT TO MEET SOMEBODY THAT GOES TO U.O.P. . . MY DAD ALWAYS TELLS ME I'VE BEEN SAVING MONEY FOR A COLLAGE, AND I JUST SAY I AM GOING TO GO TO U.O.P. WHEN I GROW UP. . .

TIM

The second feature that was noticeable in the letters was the honesty and willingness to share very important feelings with the teacher candidates. This willingness to share inner thoughts with an adult was characteristic of many of the the children. Some were more controlled and guarded, but most were as open as Lei in the following excerpts from her introductory letter and second letter in response to Dianne, her university computer pal.

HI! MY NAME IS LEI. I'M IN THE 6TH GRADE AND I HATE SCHOOL... I DON'T HAVE ANY PRIVACY, AND THERE'S ALOT OF NOISE. WHILE I'S TRING TO READ A BOOK MY SMALL BROTER IS WALING AROUND THE HOUSE SINGING THE THEME SONG FROM THE BRADY BUNCH!...

YOUR PEN PAL
LEI

Dianne responded to Lei's letter with concern about Lei's hating school. Later Dianne noted in her journal that Lei's communications all tended to be quite negative even though she had tried to prompt Lei into sharing things that she liked. In spite of Dianne's attempt to work on the positives about school, the following excerpted reply from Lei showed a deep dislike for school that is jarring:

DEAR DIANNE,

YOU MAY BE RIGHT ABOUT SCHOOL, BUT NOT

COMPLETELY. I KNOW THAT YOU NEED AN EDUCATION TO MAKE IT IN THIS WORLD, BUT I GET AN EMPTY FEELING IN MY STOMACH WHEN I THINK THAT I STILL HAVE 10 MORE BORING SCHOOL YEARS FULL OF REPORTS AND MATH UNTIL I START ENJOYING LIFE...

YOUR COMPUTER COMPANION
LEI

On the whole, this sort of negativism was unusual. The children felt comfortable in sharing their concerns about life in general and school in particular. Although the exchanges via Kidmail had not been planned as a part of the project, they became a valuable piece in the communication process. The children and the teacher candidates shared an average of three Kidmail letters during the project. They had shared two letters prior to going on-line for their first real conversation. Looking back, we would recommend that anyone planning a project that includes on-line communication to provide a similar component that allows the students to get acquainted with each other prior to on-line experiences. The letters through Kidmail not only piqued the students' interest about one another, but laid a common foundation between them. They were real to each other. They had shared experiences, thoughts, feelings. With this build-up, the on-line communication became an ultimate verification of each others' reality.

On-Line...Finally

Sign-ups to go on-line with a computer pal were posted for the first and second weeks in December. The elementary students could communicate with their computer pals between 11am and 1pm during the time allotted for reading in their classroom. A schedule was made to allow for 20 minute conversations. Teacher candidates were asked to come 10 minutes prior to their actual on-line time in order to observe the process prior to sitting at the keyboard. At both schools, the atmosphere was electric with excitement. We, as instructors, knew that this was going to be a very special experience for all involved.

Both the elementary and university students were briefed ahead of time on the etiquette of telecommunications and the framework for the conversation. Although the students would have been happy to have an informal "chat" with their computer pals, we instructors wanted to provide a more thematic learning experience. Thus, the elementary students were asked to think of three words of friendship that they could share with their computer pal when on-line. The teacher candidates were told to expect the children to volunteer the three words or, if necessary, prompt them for the words. Both were told to sign on by introducing themselves. In response to the three words, the university student would ask the elementary student to choose one, perhaps their favorite or most important, and discuss why that word was chosen. This activity provided some interesting and at times surprising insights into what the children valued in friendship. Some children showed great maturity and insight in their responses. In the following example, the maturity of thought is juxtaposed with the "bulletin board speak" Candi utilized in her conversation with her university pal, Kathryn, about the words "honesty" and "understanding".

Kathryn: DO YOU HAVE YOUR THREE WORDS OF FRIENDSHIP FOR ME?

Candi: YES I DO. THEY R: HONEST, CHEEFRUL, & UNDERSTANDING.

Kathryn: WHICH ONE DO YOU THINK IS MOST IMPORTANT IN FRIENDSHIP?

Candi: WELL, I CAN'T REALLY CHOOSE FROM THE 3, BUT I THINK THAT HONESTY AND UNDERSTANDING R THE MOST IMPORTANT 2 ME. I THINK THEY R 'CAUSE WHAT GOOD I HAVING A FRIEND IF THE'RE NOT HONEST AND UNDERSTANDING?

Kathryn: DO YOU HAVE ONE SPECIAL FRIEND?

Candi: SOFT OF. I HAVE A COUPLE OF THEM 2 SHARE MY FEELINGS WITH. I DON'T LIKE HAVE 1 SPECIAL FRIEND BECAUSE IT WOULDN'T BE FAIR 2 HAVE ALL YOUR ATTENTION 2 1 AND NOT ATTENTION 2 YOUR OTHER FRENZ.

The "bulletin board" style of Candi's writing was extremely distracting for the university student, Kathryn. She commented several times to me how difficult it was for her to tolerate this writing style and not request that Candi use standard spellings. This gave an opportunity for me, as an instructor, to initiate a discussion of idiolect and register and how they may apply in this writing environment. Although I am not sure Kathryn truly appreciated Candi's form of expression, she did accept it's function.

The teacher candidates and I were at times amazed with the students's knowledge and interests. Other times we were just baffled. At times when the communications were not clear, it gave the university student the opportunity to explore how to ask questions of students in order to elicit extended responses. One of the best examples of this was the exchange between Gina, the university student and Tim her elementary computer pal.

Gina: HELLO, JIM. DO YOU HAVE THREE WORDS OF FRIENDSHIP FOR ME?

Tim: YES, MY WORDS OF FREINDSHIP ARE FUNNY, NEAT AND KIND. FRIENDSHIP SORRY!

Gina: CAN YOU TELL ME WHAT IT MEANS WHEN A PERSON IS NEAT?

Tim: NEAT MEAN THAT PERSON IS INTERAESTING.

Gina: WHAT IS IT THAT MAKES A PERSON INTERESTING?

Tim: WHAT MAKES A PERSON NEAT TO ME IS THEY GET GOOD GRADES AND THEY HAVE A NEAT ANSESTRIAL PASTIME.

Gina: JIM, I AM VERY INTERESTED IN WHAT YOU ARE TELLING ME, BUT YOU WILL HAVE TO HELP ME LEARN ABOUT SOMETHING YOU JUST SAID. COULD YOU TELL ME ABOUT ANSESTRIAL PASTIME?

Tim: AN ANSESTRIAL PASTIME IS LIKE WHO ARE THEIR ANSESTERS.

Gina: I SEE. YOU MEAN LIKE I HAVE GREAT-

GRANDPARENTS THAT CAME FROM PORTUGAL A LONG TIME AGO.

Tim: WOW! YOU HAVE ANSESTERS FROM PORTUGAL, I HAVE ANSESTERS FROM IRELAND AND POLAND.

This exchange was enlightening not only for the rather uncommon interest expressed by this sixth grader, but for the skill in questioning I was able to observe in the teacher candidate. Her focus was correctly placed on the content, what she could learn from the student, and not the spelling. I was pleased to see this response from the teacher candidate since helping them to understand the concept of spelling development and invented spelling was an important goal for my course. Had the conversation continued, (time was up at this point) the teacher candidate could have modeled the correct spelling in the conversation, with no particular attention drawn to it.

Unfortunately, most students had time for only two on-line conversations before final exams and Christmas vacation interceded. The teacher candidates eventually met their computer pals at a Christmas party at Hazelton School. This was an important part of the project to bring a true sense of closure to the experience. The computer pals were very curious about what each other looked like, as well as just having a strong desire for actually meeting a person with whom they had shared so much, but whose voice they had never heard. Although this sort of need is probably common with pen pals, I believe that the real-time, yet silent conversations that were carried on over the modem provided a stronger sense of connection than if only letters were exchanged.

Conclusions

Was this project successful? My overall response is a resounding **"Yes!"**, in spite of the problems, delays, and the resulting lack of actual on-line time. But specifically, did we meet our goals? Looking back at the goals that were set at the beginning of the project, what was accomplished?

1. **To determine the feasibility and utility of using telecommunications in a teacher training program.** Yes, telecommunications in a teacher training program is feasible, but not without major effort on the part of the participants. The effort is not just in learning the technology, but in the very basics of facilities. Most schools do not have phone lines available for such uses in a classroom or even a computer lab. A phone line is usually not an enhancement that comes automatically upon request. Even principals who are are supportive of the concept of telecommunications, may not have the money or authorization to pull in a telephone line and purchase the modem. A project such as this one which involved 17 teacher candidates and 31 elementary students greatly taxed the two modems at each site. Scheduling involved a significant time commitment on both sides. In order to get all elementary students to respond on Kidmail and have their on-line time, the teacher had to be flexible and allow this project to supersede other subjects. On the side of the university, the teacher candidates and myself had to juggle times around classes, supervision assignments, work, and family. The bottom line, however, was commitment and we all made it work.

2. **To examine the impact of the "computer pal" idea on elementary students' writing.** We did not do a formal assessment of writing prior to the start of this project. We decided that a more qualitative approach to analyzing the characteristics of the writing and observing the student's attitude and participation would be sufficient at this stage. In terms of the writing, we can make no statements that the quality was effected. The desire to write as a result of being involved in the program was undeniable. The elementary students needed no prompts in the initial stages to get their mail off to their computer pal. This was a relatively short and novel experience, so if the enthusiasm would have maintained over a longer period of time is not known. The on-line conversations were so exciting and new to the students, that they hardly noticed that they were writing. They were challenged in spelling and grammar to be able to clearly communicate to their computer pal. Some showed a real awareness of this by apologizing in writing for being a "bad speller" or silently by the repeated attempts at spelling a word. Others such as Tim, let the meaning speak louder than the spelling, showing his priorities were correctly placed in the communication. The project gave the teacher candidates the opportunity to look at a student's writing from a developmental and situational viewpoint. The writing process became almost transparent as the students typed on-line. The corrections, false starts, and regressions allowed the students to almost see how the student composed thoughts into a graphic representation. It was an excellent means to compare how the writing process can actually parallel the reading process. The only global hang-up which would have facilitated the entire process was the lack of keyboarding skill on the part of the elementary students. Fortunately all of the teacher candidates were accomplished keyboarders. If computer based writing is to be effective in the curriculum, keyboarding is a necessity. The slow keyboarding, compounded with limited accessibility to computers, were two major and interrelated problems in this project because of the time pressures faced in schools.

3. **To examine what teacher candidates would learn about children and their writing.** As each teacher candidate brought different experiences to the project, so did they walk away with different understandings and learnings based on the nature of the interaction with their computer pals. Spelling, grammar, sentence complexity were mentioned by several teacher candidates, both positively and negatively as would be expected. Those teacher candidates who had two computer pals frequently

commented on how different the skill levels, point of view, style, and perspective were between the two. Through the writing, the teacher candidates gained valuable insights into the world of fifth and sixth graders. In their written evaluations, the teacher candidates commented on what they learned about the language and culture of their young friends. They found out that sixth graders are "very involved with friends from school" and "concerned about their appearance." They expressed some real surprise at the worldliness of the students, "...some 6th graders are getting interested in the opposite sex already!" and "...they are up on current events (and not just in the fashion industry)." One 22 year-old teacher candidate made the following observation, tying this experience to her recollections of the past:

> It seems that these kids are more mature as compared to when I was in 6th grade....Yes we used our own language so that adults wouldn't understand, but it seemed so simple then. The sayings and slangs of today are deeper and have more thought involved, for example, "later days, when our paths blow together again!" Now this is deep. Since when do 5th and 6th graders use forms of poetry for expression?

4. **To develop positive attitudes on the part of teacher candidates toward the use of computers in education.** All of the teacher candidates had little or no experience with computers. None had ever used telecommunications. When asked if this experience influenced their desire to have computers in their classroom, I was surprised only in the the unbridled enthusiasm. Some of the teacher candidates can speak for themselves:

> Now I hope my classroom has computers! I really liked this project. Besides learning about 6th graders; overall language and writing skills, this project helped me to remember what it was once like to be in the 6th grade and to understand more about what children are thinking and feeling--because sometimes even we teachers forget.

> I want to use computers. I had always wanted to work with computers, but now I know I can use computers with great skill.

> I feel the use of the computers in the classroom is essential in today's education. ... In this case the computer was not used as the ultimate goal- (*of just*) learning to use a computer, but as a means to an end--(*that is*) improving writing and communication skills.

And finally, from "doubting Thomas" to true believer,

> When I first thought about using a computer in the classroom, I didn't like the idea. I feel that students should learn how to do the basics manually before using the computer... Then I participated in this project and I have seen how much more a computer can provide in the area of writing. I would like to have a telecommunication system set up in my classroom to do the same type of project we did here...

With greater knowledge of modems and communications software, the software and management bugs mostly worked out, and a plan to do shared book conversations on-line; the next semester offered a new and exciting challenge for my students and me.

* FrEd Sender and Kidmail are public domain software available through Computer Using Educators, Softswap.

References

Tennis, J. " Hooking Up a Modem." *Electronic Learning,* 5 (4), pp.14-15.

Tennis, J. " Computer Club Goes On-Line." *Electronic Learning,* 5 (6), pp. 20-21.

Tennis, J. " Beaming in the Experts." *Electronic Learning,* 5 (8), pp. 16,21.

ProModem 1200 Instruction Manual. Prometheus Products, Inc., Fremont, CA: 1985.

"Special Issue--Telecommunications." *The Computing Teacher* 14 (7).

Appleworks™, a registered trademark of Claris Corporation, Mountain View Ca.

ProDos™, a registered trademark of Apple Computer, Inc.

WRITING AND RESEARCHING IN THE LANGUAGE ARTS AND SOCIAL STUDIES

Barry Jahn
Marianne Teague, Gerald V. Teague, G. Edison Weslow

ABSTRACT: C.P.R. (Computer Processed Research)

Barry Jahn

This curriculum development project involved two advanced writing classes at Sprague High School in Salem, Oregon. The primary objective was for each student to have increased exposure in utilizing the computer as a tool to accomplish a required research project. It involved the use of computers in telecommunications, outlining, word processing, spell checking, thesaurus, grammar checking, and the use of graphics software.

The participants were divided into three basic groups: those using the standard research methods in the library and a typewriter; those using the standard library research methods but word processing the outline and paper; and a third group computerizing the entire project.

All three groups were monitored as they proceeded through the research process. Prior to editing and producing the final copy, the participating teachers divided the research project into five checkpoints. We were thus able to monitor each component as the weeks progressed. Post observation discussions were also recorded.

In brief, using telecommunications software and a national database called DIALOG, the 'computer' group was able to quickly reduce their topic and locate relevant sources. Downloading most of the data quickly into their word processor saved hours of typing. They spent most of their time editing existing text and in the organization of their paper.

An immediate significant difference was observed when the 'computer' group had to deal with an 'overkill' of information—and the necessity to further 'reduce' their topic!

The pros and cons of this project will be discussed in more detail.

ABSTRACT: Enhancing the Social Studies Curriculum: Use of Video, Hypertext, and Telecommunications

Marianne Teague, Northern Middle School, Calvert County, MD; Gerald V. Teague, University of Maryland; G. Edison Weslow, Arrowhead Elementary, Prince George's County, MD.

A research and development project is being conducted by Maryland students from two schools located in different counties. The three-fold goal of the project is to prepare students to: 1) utilize electronic information retrieval systems; 2) exchange information electronically; and 3) create documents using computer technology. In addition, a linkage is being created between the classroom social studies curriculum and media center research skills instruction as well as between schools.

Videotapes of historic and prominent sites throughout the US midwest were made by several educators during a cross country trip in the summer of 1988 under the sponsorship of a Maryland Humanities Council grant. During the 1988-89 school year, sixth grade students from Northern Middle School in Calvert County were paired with sixth graders from Arrowhead Elementary School in Prince George's County. As interschool research teams, students collaboratively researched the background information on twelve taped locales such as the Grand Canyon, Carlsbad Caverns, Boot Hill, and ancient Indian cliff dwellings.

Students were taught to use CompuServe, an electronic information service, for information retrieval and communication. Grolier's Electronic Encyclopedia provided historical data on sites to supplement print materials. Modem access via CompuServe to the Washington Post newspaper allowed students to search for any current information such as the rencent fires in Yellowstone National Park. Educators and students in states with sites under study were also queried using the electronic educational forums.

The videotapes were edited by the students who provided the narrative for their specific topic from researched material. The end product will be a 30-45 minute student-narrated videotape with accompanying student word processed print guide. Using the modules developed by the students, hypertext stacks on selected sites will be created. The narrated videotape will be the basis for production of an interactive videotape.

In addition to introducing the students to telecommunications and electronic information searching, the project has taught students collaborative research strategies and use of technology for tape and print production. Additional benefits have been the relationship developed between students at a rural and a suburban school resulting in personal interest and exchange of photos and the enthusiasm of students for this exciting way of learning.

INFORMATION TECHNOLOGY AND SCIENCE EDUCATION: THE 1988 AETS YEARBOOK

James D. Ellis
Biological Sciences Curriculum Study
830 North Tejon Street, Suite 405
Colorado Springs, CO 80903

Paul Horwitz
BBN Laboratories, Inc.
Cambridge, MA

Cecilia Lenk
TERC
Cambridge, MA

ABSTRACT

As the United States changes from an industrial-based society to an information society, the goals, purposes, and methods of American education are evolving. In response to this societal and economic change, more than100 national and state reports have called for a reform in education that emphasizes the development of critical-thinking skills, problem solving, knowledge utilization, scientific and technologic literacy, and computer literacy. Because rapid advances in information technology are the driving force behind the information revolution, computer literacy is now recognized as a basic skill, together with reading, writing, and arithmetic.

Information technology and science education share the platform of the educational reform movement. Science education, redefined as education in science, technology, and society, is one plank that supports the citizens who will participate fully in a society and culture that have a scientific and technologic infrastructure. Information technology, another plank, enables us to access the enormous volumes of scientific and technologic information and to synthesize and produce useful knowledge from it. Information technology, therefore, has forced new goals for education that have students learn about information technology (what it is, what it does, and how it affects society) and become proficient in using information technology in the pursuit of knowledge to improve the effectiveness and efficiency of learning.

This yearbook examines the overlap of information technology and science education. The first five chapters present a vision of how information technology can enhance science education. Chapters six through eight present the status of information technology in science education and discuss the relationship between them. Chapters nine through fourteen discuss how we can improve the use of information technology by teachers and students in science classrooms.

ELECTRONIC STUDYING: HYPERCARD AND OTHER APPLICATIONS TO HELP STUDENTS STUDY "BETTER" NOT "HARDER"

Lynne Anderson-Inman
University of Oregon
Eugene, OR

ABSTRACT

The presentation will describe and show examples of three computer applications for improving the effectiveness and efficiency of students' independent study of content area materials. Two of these applications have been created with the Hypercard authoring language created for Macintosh computers.

The first application is a method for using computer-based outlining programs to promote successful studying of content area textbooks. Research on this method has indicated that students who are taught to outline, summarize, and self-test using a computerized outlining program comprehend significantly more of the material being studied than students using a traditional "read and re-read" study procedure. The presentation will include an overview of how to teach students to use outlining programs for studying and will provide data from recent research on the effectiveness of this study procedure.

The second application is a HyperCard teacher-authoring interface (Electrotext) for creating electronic textbooks with enhancements for vocabulary support, schema development, and improved comprehension. Students studying in an "Electrotext" environment are able to: (a) access definitional information about any word on the screen, (b) self-test their level of understanding by selecting and answering page-specific comprehension questions, (c) quickly access background information, pictures, and chronological timelines, (d) use a structured overview to understand the conceptual relationships of important technical terms, and (e) employ "study stacks" to acquire indepth understanding of key concepts. In the presentation, the Electrotext study environment will be explained and demonstrated using materials created by teachers in a variety of subject areas.

The third application is a Hypercard reference and notation system for collecting, analyzing, and synthesizing the literature on any given topic. This "hyperbibliography" allows students to enter bibliographic references for information of any format, attach study notes to any entry, and create a wide variety of links among the body of entries created. Unique to this hyperbibliography is the ability of the student to establish conceptual "threads" throughout the body of information being studied, collect entries into a "bag" for later use or analysis, and retain a sequential "trail" of the entries accessed in any given study session.

PANELISTS

John Tenny
Willamette University
Salem, OR

Mark Horney
Univeristy of Oregon
Eugene, OR

SPONSOR: SIGCUE

RESEARCH IN THE APPLICATION OF COMPUTERS IN EDUCATION: TRENDS AND ISSUES

Betty Collis
University of Twente
Enschede, The Netherlands

ABSTRACT

For the last four years, ICCE (The International Council for Computers in Education) has included a column called "Research Windows" in its monthly publication, *The Computing Teacher*. The purpose of this column has been to identify well-done research studies that have particular relevance to the classroom teacher. What are some of the trends that emerge through these four years of research summaries? How can these trends inform the teacher who is using computers in his or her classroom?

Certain trends can clearly be identified. Some of these are:

1. Regular confirmation of the importance of the teacher with respect to computer applications in the classroom. Also, indication that, at the present time at least, computers are not replacing teachers or even making teachers' tasks easier, but instead offer teachers new challenges and opportunities.

2. Awareness that computer impact is never a simple situation but instead will vary from situation to situation and also among students in a particular situation. In other words, there is no "right way" or clear conclusion that can be made about aspects of computer use, other than acknowledging their complexity.

3. Inability to confirm experiences with computer programming as having a generalizable impact on higher-level thinking skills or problem solving skills.

4. Continued interest in word processing as an instructional tool, but also awareness (see Points 1 and 2) that the impact of word processing experiences are strongly related to the way the teacher embeds these experiences in a larger instructional setting.

5. Encouraging support for the value of MBLs (microcomputer-based laboratories).

These, and other trends will be summarized and examples of studies relating to the trends will be given.

Discussion of the pertinence of the research to the classroom teacher will be an ongoing part of the presentation.

SPONSOR: ICCE

COMPUTERS AND LIMITED-ENGLISH PROFICIENT STUDENTS: TOOLS FOR SUPPORTING SCIENCE LEARNING IN MIDDLE AND HIGH SCHOOL

Ann S. Rosebery, Beth Warren, Bertram C. Bruce
Bolt, Beranek and Newman, Inc.
10 Moulton Street
Cambridge, MA 02138

ABSTRACT

Students of limited English proficiency (LEP) are receiving substandard education in science and mathematics. Their levels of achievement are among the lowest in the nation. Their participation in undergraduate and graduate study in these areas is minimal. Not surprisingly, they are poorly represented in scientific and mathematical professions like engineering, medicine, and teaching (Melendez, 1989; Reeves,1988; Mullis and Jenkins, 1988; Steen 1988).

To begin to address this problem, the Cheche Konnen Project[1] is working with teachers to develop an investigation-based approach to science and mathematics for LEP students, the majority of whom have had little or no exposure to science or, more importantly, to scientific ways of thinking. The approach emphasizes not only the reasoning processes and conceptual knowledge that fuel the activities of science and mathematics but also the social and linguistic processes that mediate them. What this means in practice is that students engage actively and collaboratively in scientific inquiry in much the same way that practicing scientists do. They define problems; generate hypotheses; plan and design research; collect, analyze, and interpret data; write laboratory and field notes; and make decisions based on their research. Language, both English and students' native languages, plays a crucial role in the approach as a system both for thinking and talking about science and mathematics and for communicating and sharing ideas.

The computer is an important component in this approach. Students, none of whom has ever used a computer before, use technology in many of the same ways that practicing scientists use them: as tools that support inquiry and understanding. Word processors are the tool of choice for language-mediated activities such as preparing data collection sheets, composing interview questionnaires, writing letters to local experts, and publishing research findings. Statistics and graphics packages are used to analyze data, for example, to describe differences in the pH of water samples and to analyze the results of a blind taste test of water from different school fountains. HyperCard is used to create students' databases which are compiled into a field guide for a local pond. Students design cards to hold specific and detailed information about the ecological habitat of their site. They create drawings of microscopic (e.g., protozoa, plankton) and macroscopic (e.g., plants insects, fish) life observed in and around the water. With special tools incorporated into HyperCard, they measure and scale their drawings to accurately represent the observed plant or animal.

While any computer can be used in this approach, we have found the Apple Macintosh to be particularly well-suited to meet the needs of LEP students. For example, because the interface is in large part icon based, acquiring computer expertise does not depend on expertise in English. This means that LEP students can become skilled users as quickly and easily as native speakers of English. Second, the wide range of fonts that are standard on the Mac allow students to compose easily in their native languages as well as in English. This encourages students to read and write during their investigations. Third, the Mac's superior graphics capability can support ways of thinking and problem solving that are relatively language-independent (e.g., interactive graphs, measuring, scaling, drawing). This means that students can engage in problem solving that is conceptually appropriate to their intellectual abilities rather than problem solving that is limited artificially by their proficiency in English.

Cheche Konnen is being field tested in a public middle school and a public high school. Three teachers and approximately ninety students are participating in the project. Students represent a range of linguistic, cultural, and academic backgrounds. Their countries of origin include Haiti, the Dominican Republic, El Salvador, Cape Verde, Brazil, Ethiopia, and Eritrea. Their academic skill ranges from those who are barely literate in their native language to those who can read write and do mathematics on an upper elementary level.

References

Melendez, Sara. (1989) "Linguistic minorities in science and mathematics." Presentation to Mid-Atlantic Conference on Equity, Prince George's County, MD.

Mullis, I. & Jenkins, L. (1988). *The science report card: Elements of risk and recovery.* Princeton, NJ: ETS.

Reeves, S.M. (1989). "Self-interst and the common weal: Focusing on the bottom half." *Education Week*, April 17, 14-21.

Steen, L.A. (1988). "Mathematics and education: A predictor of scientific competitiveness." *Science*, 237, 251-252 & 302.

---

[1] This project is one of four being funded under the Innovative Approaches Research Project by the Office of Bilingual Education and Minority Language Affairs and administered by Development Associates, Inc. It runs through July, 1990. Cheche Konnen means "searching for knowledge" in Haitian Creole.

APPROACHES TO PREPARING EDUCATIONAL SOFTWARE DESIGNERS

Shelly Heller, Chair
Department of Electrical Engineering and Computer Science
School of Engineering and Applied Science
George Washington University
Washington, D.C. 20052

ABSTRACT

The quest for good educational software most often starts with an evaluation of existing products in an attempt to select those programs which are worthy of being included in the classroom. While most educators agree that good educational software should be developed with the classroom context in mind and should be the cooperative effort of educators, designers, and programmers, most courseware is not developed in that fashion. Very often the existing software is designed and implemented by individuals whose training comes from areas of expertise outside of education or computing. The computer environment is not like a textbook or a film yet many of the designers are creating software based on their knowledge of textbook or film design, in part because opportunities to learn about the unique capabilities of the computer have not been available. Only recently have graduate schools of computer science or educational technology begun to address this issue.

This special session will focus on different approaches in courses which are directed at educational software designers and implementors. Panel members will discuss various techniques for preparing individuals to write and produce educational software. These approaches include, but are not limited to:

- courses which are cross-listed and are directed at educators and computer scientists;
- courses which provide certification as a proficient educational software designer;
- courses which are founded in traditional human factors design;
- opportunities which are addressed at the teacher-educator;
- experiences which are directed at high-school students.

PANELISTS

Bengt Bengtsson
Ministry of Education
Sweden

Janet Graeber
University of San Francisco

Dianne Martin
George Washington University

Teresa Rosegrant
George Washington University

Renate Schulz-Zander
Institute of Science Education
Kiel, Germany

LEARNING STYLES OF COMPUTER USERS

Danielle R. Berstein
Mathematics/Computer Science Department
Kean College of New Jersey
Metuchen, NJ

Abstract

The availability of microcomputers and an abundance of software packages have combined to give non-computer professionals powerful problem-solving capabilities. How does this new class of computer users learn and cope with the proliferation of software?

In this study, the relationship between learning styles and achievement in understanding a software package was examined. Twenty-one management science students in a state college were taught to use a spreadsheet package. Their preferred learning style was assessed by administering Kolb's Learning Style Inventory and by asking them whether they like to learn by trial and error (in a procedural manner) or by reading and understanding the package first (in a conceptual manner). The students were also tested in their use and understanding of spreadsheets.

Results showed that students who preferred to learn in a procedural manner did better in every outcome measure, though not significantly better in all these measures. In a future research project, the relationship between training methods and individual learning styles of computer users will be studied.

Introduction

The availability of microcomputers and an abundance of software packages have combined to give non-computer professionals powerful problem-solving capabilities. How do people cope with this proliferation of software? Certainly, individual differences of users may account for their performance with these packages but the training they receive may also affect the success and confidence with which they use software.

There has also been a change in the types of computer users. Until recently, computer users could be divided into two groups: educated and experienced computer professionals who understood and appreciated the internal workings of the machine and trained clerical workers who operated a computer system as part of a routine task. A new category of user, the "power user," has developed--people who have found that computers can facilitate the routine aspect of their jobs. They are not necessarily as interested in the intricacies of the hardware or software as are the computer professionals. They learn how to use software as part of their job and often run several applications in an average day. Typically, they are neither afraid of the computer as a naive user might be nor enthralled by it as a stereotypic hacker. Power users are found among middle managers, accountants, and information scientists.

There have been few studies on the learning approaches of power users. If more were known, better training might result. In advertising software packages, the message is how quickly and easily one can use them without much background. Instruction manuals do not offer a conceptual understanding or mental model of the function and purpose of the software. Yet it has been found that it is important that users build a mental model, or else they get lost (Borgman,1983).

This paper describes a pilot study which looked at how management science students learned a new software package. The goal of the research was to study the relationship between learning style and outcome in computer usage. The study is part of a larger research project which will investigate how learning styles interact with training methods.

Previous Work

Learning style is the individual way in which a person gains knowledge and understanding. Gregorc (1979) defines a learning style as consisting of distinctive behaviors which serve as indicators of how a person learns from and adapts to the environment. Learning style also gives a clue as to how a person's mind operates.

Kolb's learning style model (Kolb, 1974, 1984) is based on experiential learning theory which propounds that people learn from experience, from their everyday activities as well as from concepts and books. Kolb's Learning Style Inventory (L.S.I.) was developed to measure a person's dominant learning mode. Kolb's L.S.I. is a self-report instrument based on a rank ordering of 4 possible phrases which best finish a sentence describing a learning style.

Kolb identified four types of abilities which are necessary to facilitate learning (Smith & Kolb, 1986): concrete experience, reflective observation, abstract conceptualization and active experimentation. **Concrete experience** is the ability to be involved in experiences and to deal with immediate human situations in a personal way. **Reflective observation** focuses on understanding the meaning of ideas and situations by carefully observing experiences from many points of view and reflecting on them. **Abstract conceptualization** is the capacity to use logic, ideas and concepts and emphasizes thinking and integrating the observations into logical theories. **Active experimentation** concentrates on actively influencing people and changing situations. The theories are used to solve problems and make decisions.

The extent to which a person emphasizes abstract conceptualization (AC) over concrete experience (CE) and the extent to which a person emphasizes active experimentation (AE) over reflective observation (RO) constitutes an individual's predominant learning style. Based on the interaction among the four abilities (Figure 1), Kolb identifies four learning styles.

Divergers emphasize concrete experience and reflective observation and are able to view concrete situations from many perspectives. Many people with liberal arts and humanities background are in this category. **Assimilators** have abstract conceptualization and reflective observation as their dominant learning abilities. They can create theoretical models and are less concerned with the practical use of theories than in the preciseness and soundness of the theory. This learning style is found in basic scientists and mathematicians. **Convergers** are strong on abstract conceptualization and active experimentation. They like practical application of ideas and do best where there is a single correct answer. Engineers typically fall into this group. Accommodators are best at concrete experience and active experimentation, and they can adapt themselves to specific immediate circumstances. Accommodators take on action-oriented jobs such as marketing or sales.

Learning style studies are typically correlational studies in which the goal is to describe the members of a particular occupational group or college major. Studies have been conducted with on-line searchers (Bellardo, 1985; Borgman, 1985, 1986; Fidel, 1984; Woelfl, 1984) and data processing professionals (Bush and Schkade, 1985; Lyons, 1985; Sitton and Chmelir, 1984). These studies characterize users according to their learning styles and hypothesize that individual characteristics as measured by a learning style inventory correlate with success.

Methodology

Subjects

Twenty-one college students (seven women and fourteen men) majoring in management science participated in the study. These students were enrolled in a computer science course offered at a New Jersey state college. The course, the second in a sequence of computer science courses designed for management science majors, teaches the use of popular software packages (word processing, spreadsheets and database management systems) that business students are likely to need. The students are assumed to have the fundamentals of computer literacy from a previous course and are not considered novices. They see the need for software packages in their future college work and in subsequent employment and regard themselves as potential power users.

Environment

The students learned to use a spreadsheet package with a combination of lecture, discussion and hands-on exercises. A spreadsheet is a popular type of software package designed to implement business applications. Spreadsheets are powerful tools that enable users to do much of the work that formerly required programming in a procedural language. They offer direct manipulation of an accountant's model of reality and provide a variety of powerful operations in a relatively simple manner (Shneiderman, 1986). The software package has enough features of varying complexity to allow for diverse levels of use. Yet, the resultant spreadsheet can be judged as either fulfilling or not fulfilling the requirements of the problem it is trying to solve.

Because a spreadsheet is menu-driven, all its commands are visible and available. There is little to memorize. A command can be specified by keying one letter or function key and little typing skill is needed. A context-sensitive HELP feature can be employed at any level in the hierarchy to explain the use of a command. However, the online HELP does not describe how commands interact.

Experimental Tasks

The experimental tasks consisted of an online test and an essay question. The students created a spreadsheet under test conditions. The essay question asked the students to write on the purposes and functions of spreadsheets. The contents of the answers were analyzed and classified as to the extent to which they displayed evidence of a mental model.

Measures

Two separate instruments measured learning style. The first learning measure (LS) used Kolb's Learning Style Inventory which classified subjects into divergers, assimilators, convergers or accommodators. The second learning style measure (CULS) was obtained by asking students to rate themselves on a seven-point scale on whether they preferred to learn new software procedurally (by getting on the computer, following instructions and experimenting) or conceptually (by reading, lectures and understanding the package first). See Figure 2.

The outcome variables were gathered as a result of an online test and an essay question. The approach measures were number of keystrokes (NK) and the number of times a false path was taken through the spreadsheet menu (ERR). They were obtained by computer monitoring during the online spreadsheet test. The performance variable was the mark for the online test (TMK) based on correctness and flexibility of the spreadsheet. Evidence of a mental model (MM) was obtained by scoring the essay.

Analysis and Results

The 21 students varied in their preferred learning styles. There were 5 divergers, 7 assimilators, 4 convergers and 5 accommodators. Based on their CULS scores, 7 were classified as conceptual and 14 as procedural. The correlation results (Table 1) shows that there is a significant relationship between Kolb's LS and CULS.

The procedural learners tended toward the converger and accommodator styles while the conceptual learners were mostly in the diverger and assimilator class. There was a significant positive correlation between the CULS score and the mark on the test (the higher CULS number indicating a preference towards procedural learning) as well as a significant positive correlation between the number of keystrokes and errors.

Analysis of variance (ANOV) was performed in two ways: by grouping the subjects based on Kolb's four learning styles (LS) and by dichotomizing the sample into procedural and conceptual learners based on CULS. The number of subjects in each group in either case was not equal. The ANOV results based on Kolb's learning style measures did not show any significant difference in any outcome measure. There were no discernible patterns or trends. Using the dichotomized sample (procedural and conceptual), the ANOV results shown in table 2 indicate that the procedural learners did better (though not always significantly better) than the conceptual learners in every outcome measure.

Discussion

The purpose of the study was to determine if there was a relationship between learning style and outcome in computer use. Learning style was measured in two ways--Kolb's Learning Style Inventory (LS) and CULS. The students fell in all four of Kolb's learning style groups suggesting that business majors are now a diverse group.

The larger number of procedural learners may be explained in several ways. Software packages, especially a menu-driven interface, encourage a procedural approach. The advertising in trade magazines try to convince power users that they do not have to understand much about computing. They just learn by doing. The students' only experience with computers may have been with software packages except for a look at BASIC programming, an experience best forgotten. Secondarily, the classroom teaching may have emphasized procedural learning over conceptual learning by encouraging students to use certain keys and "see what happens." The learning style instruments were administered toward the end of the course when the students had been fully exposed to several software packages and the instructor's teaching style. The results may have been different if the learning style approaches had been captured at the beginning of the course.

The above discussion may also help explain the ANOV results. The procedural group did better (though not significantly better) than the conceptual group in every measure. The procedural learners used fewer keystrokes, made fewer errors, received a better mark on the test and developed a fuller mental model than the conceptual learners. Procedural learners, almost by definition, spend more time on the machine than conceptual learners. That may be what is needed to learn software packages.

The next study will look at the interaction of learning style with training method. Two classes, taught by the same instructor, will use two training approaches: a conceptual approach and a procedural approach. This will enable us to look at the interaction between learning styles and training methods.

References

Bellardo, T. (1985). An investigation of online searcher traits and their relationship to search outcome. *Journal of the American Society for Information Science*, 36(4), 241-250.

Borgman, C.L. (1983). *The user's mental model of an information retrieval system: Effects on performance*. Ph.D. dissertation, Stanford University.

Borgman, C.L. (1985). Individual differences in the use of technology: Work in progress. *Proceedings of the American Society for Information Science*, 22, 243-249.

Borgman, C.L. (1986). Individual differences in the use of information retrieval systems: A pilot study. *Proceedings of the American Society for Information Science*, 23, 20-31.

Bush, C. & Schkade, L. (1985). In search of the perfect programmer. *Datamation*, 31(6), 128-132.

Fidel, R. (1984). Online searching styles: A case-study-based model of searching behavior. *Journal of the American Society for Information Science*, 35(4), 211-221.

Gregorc, A.F. (1979). Learning/teaching styles: Potent forces behind them. *Educational Leadership*, 36, 234-236.

Kolb, D.A. (1974). On management and the learning process. In D.A. Kolb, I.M. Rubin and J.M. McIntyre (Eds.). *Organizational psychology: A book of readings*. (pp. 27-42). Englewood Cliffs, NJ: Prentice-Hall.

Kolb, D. (1984). *Experiential Learning*. Englewood Cliffs, NJ: Prentice-Hall.

Kolb, D.A. (1985). *Learning Style Inventory*. Boston: McBer.

Lyons, M.L. (1985). The DP psyche. *Datamation*, 31(16), 103-110.

Shneiderman, B. (1986). Empirical studies of programmers: The territory, paths, and destinations. In E. Soloway and S. Iyengar (Eds.) *Empirical Studies of Programmers*. (pp. 1-12). Norwood, NJ: Ablex.

Sitton, S. & Chmelir, G. (1984). The intuitive computer programmer. *Datamation*, 30(16), 137-140.

Smith, D. & Kolb, D. (1986). *User's guide for the learning style inventory*. Boston: McBer.

Woelfl, N.N. (1984). *Individual differences in online search behavior: The effects of learning style and cognitive abilities on process and outcome* Doctoral dissertation, Case Western Reserve.

Figure 1. Kolb's Learning Abilities and Learning Styles

```
                       CONCRETE EXPERIENCE (CE)
                                |
                                |
                                |
   ACCOMMODATOR                 |        DIVERGER
                                |
                                |
                                |
ACTIVE                          |                    REFLECTIVE
----------------------------------------------------------------
EXPERIMENTATION (AE)            |              OBSERVATION (RO)
                                |
                                |
                                |
    CONVERGER                   |        ASSIMILATOR
                                |
                                |
                                |
                                |
                                |
                 ABSTRACT CONCEPTUALIZATION (AC)

----------------------------------------------------------------
```

Figure 2. A Measure of a Computer User's Learning Style (CULS)

People like to learn about computing in different ways.

- Some people read about a software package or they listen to a lecture on it and really try to understand it before they start out on the computer.
- Others want to get on the machine with minimal knowledge and want to see the results of trial anderror.

How do you like to start learning a new software package?

```
|--------|--------|--------|--------|--------|--------|--------|
   1         2        3        4        5        6        7

Understand it first                              Trial and error
```

TABLE 1 . Correlation of learning styles and outcome measures

	LS	CULS	NK	ERR	TMK	MM
LS	1.0	0.55*	-0.35	-0.07	0.29	0.09
CULS		1.0	-0.28	-0.18	0.43*	0.27
NK			1.0	0.52*	-0.29	-0.25
ERR				1.0	-0.03	-0.13
TMK					1.0	0.30
MM						1.0

*P = .05

LS	- One of four learning styles based on Kolb's L.S.I.
CULS	- Response to question shown in Figure 2
NK	- Number of keystrokes
ERR	- Number of wrong paths taken
TMK	- Mark on online spreadsheet test
MM	- A measure of completeness of a mental model based on scoring the essay.

Table 2. Means and ANOVA comparing conceptual and procedural learners

	CONCEPTUAL		PROCEDURAL			
	MEAN	SD	MEAN	SD	F	p
NK	1251.29	657.54	741.86	221.40	7.12	0.015
ERR	26.57	24.86	9.93	14.59	3.79	0.066
TMK	28.86	5.52	32.79	3.62	3.87	0.064
MM	1.43	1.27	2.50	1.34	3.06	0.096

LEARNING TO USE versus USING TO LEARN

Patrick Casey
Nancy Cunniff
Robert P. Taylor

Center for Intelligent Tools in Education
Teachers College * Columbia University
Box 8
New York, NY 10027

Abstract

This paper reports research designed to determine how new users of a specific computer-based instructional application environment can be most efficiently trained in the standard use of that environment. It assumes that learner time and effort spent *learning how to use* a specific piece of software should be absolutely minimized, so learners can quickly and correctly get on with *using* it for the purpose for which it was designed.

The research involves an experiment with three different treatment groups learning to use the same software environment. All were given the same training task to complete and the same print description of how to perform it. The first attempted to perform it via a *scenario machine* which simulated the actual software environment but constrained the user to a single prescribed path representing standard, elementary use of the software; the second, via a modified *training wheels* version of the actual software environment, that allowed some digressive explorations and errors but prevented others; and the third, via the actual software environment, that allowed them to digress at will, along any optional path available to any normal user of the software.

Measurements were made of each subject's total time to task completion and total keystroke production : (1) in training and, subsequent to training, in both (2) a near and (3) a far transfer task. By both measures, the tightly controlled *scenario machine* approach proved superior.

Introduction

One of the ironies in using a computer environment of any kind to support instructional activity is that learner time and energy must first be expended on the particular environment itself, before the learner can use that environment to support his or her learning of those concepts and procedures which are the real focus of the instructional activity. Obviously educators determined to use the computer to support learning must develop ways to minimize the time and energy needed for such preliminary, and in many ways, extraneous activity. Otherwise, more and more learner time will be absorbed in the prerequisite familiarization period required for the successful use of each different piece of application software for word processing, programming, spread sheeting, database manipulation, animation, telecommunication, and so on. This paper assumes that research into how best to accomplish this minimization is appropriate, and reports on some research of this type that is currently underway at Teachers College, Columbia University.

The computer environment involved is one designed to support the learning and use of a graphically represented, procedural programming language, FPL (Taylor, 1982; Taylor, Cunniff & Uchiyama, 1986), used in a number of introductory programming classes each year. The research involves determining which of several approaches may be most efficient in moving the new student from total unfamiliarity with the environment of this programming language to one where he or she can comfortably proceed with the real business the environment is supposed to support: learning to program in the visual programming language. Though the research reported here deals with novices learning to use a programming language, it is a specific instance of the general interface-familiarization problem. Its implications are therefore quite general and not peculiar to programming.

Relevant Research on Learning to Use Software Environments

> Restricting vocabulary to a small number of familiar, consistently used terms is essential to begin developing the user's knowledge of the system. (Shneiderman, 1987, p. 54)

The problems involved in training people to use complex computer systems have been well documented in the literature. Problems typically arise when a new user is presented with a large number of new, unfamiliar terms and procedures and is required to use that vocabulary in the navigation of an unfamiliar system. The solution to this dilemma, as often as not, requires training with a restricted vocabulary and limiting the range of functions until the new user gains confidence and basic familiarity with the system.

In response to this training problem, Carroll (1984) advocated a minimalist design for training materials. Minimalist design aims at directing and supporting the natural learning styles and strategies of users. In essence, this plan advocates drastically shortening instruction by eliminating all nonessential information, and recommends immediately involving the learner in realistic tasks (Black, Carroll, and McGuigan, 1987). This work has taken two forms, the production of minimal *manuals* and the development of *modified training* interfaces. A minimal manual presents the learner with print document of the essential information needed to use a system. This information can be explicitly stated or the user can be guided by the manual to infer essential system information.

Two approaches have been developed to the modified training interface: (1) the "training wheels" approach and (2) the "scenario machine" approach.

Training Wheels

As the name implies, a training wheels system provides the new user with an interface that has features built in to help prevent problems. The novice learns on an initially simple and error-cushioned system interface that helps develop foundational concepts and confidence, and a bridge from mastery of the basic functions to acquaintance with the complete system. The hardware, menus, messages, and prompts used are identical to those on the full system so that transition from training to full use is smooth (Carroll & Carrithers, 1984A). Training is facilitated because: (1) it reduces learner frustration by diminishing error consequences; (2) it directs error recovery by providing explicit methods for resolving the errors; and (3) it supports hypothesis testing by limiting the space of accessible system states (Carroll & Kay, 1985). The rationale is that a training wheels interface provides an exploratory environment (Carroll, 1982) for the new user that permits active involvement in the learning process but still provides reasonable protection from the digressive consequences of errors that active learning inevitably entails (Carroll & Mack, 1984).

Carroll and Carrithers (1984B) studied new users attempting to learn the basic functions on a word processing system, producing as one product of their study an inventory of new-user errors. They hypothesized that blocking common errors during training would eliminate some of the frustration and confusion caused in the early stages of learning, and thereby facilitate novices' learning of the system. They tested this hypothesis on two groups of subjects. One used a training wheels interface where the major errors were blocked, the other group used a full, unaltered system. The results revealed that subjects using the training wheels interface took less time to complete the training and were more successful in completing the assigned task. On subsequent tests, the training wheels group fared better than the group trained on the full system.

Scenario Machine

Young and Barnard (1987) refer to a single instance of human-computer interaction as a "scenario." This concept of a computer scenario was developed into the idea of a "scenario machine" -- a limiting case of a training wheels approach, the second approach to the modified the training interface. Carroll & Kay (1985) discuss the scenario machine as an online presentation of a complete, guided series of interactions to the user. During training, from beginning to end, only a single scenario can be run: the user is forced along a prescribed path with no choices. To be an effective learning environment, the designated scenario should represent the most basic and representative functions of the full system, and should *appear* identical with that of the actual system. The inputs to which the scenario machine will actively respond are limited to those that move system action along the pre-determined path of the training scenario. Exploration and creative additions, and/or modifications to the task specified for training are blocked. Thus, the user can trace one and only one path, as s/he performs the requirements of the exercise.

Carroll and Kay (1985) studied the use of a scenario machine user interface. Their study contrasted four groups of subjects each using a different scenario machine design with a control group being trained on a full system. *The scenario machine design that most attenuated user errors produced the best training results.* They also tested designs that provided the learner with action prompts and error correction feedback during training. Such modifications, however, proved inferior to a system that simply ignored incorrect user input. Mayer's earlier conclusion (1980), that forcing self-generated elaboration during learning improves later performance, may explain the superiority of a scenario system that forces the user to determine the reason why input is not acceptable by the computer. Carroll and Kay conclude that the scenario machine appears to provide a training edge over full system use even if this advantage is only to facilitate "getting started."

Context for the Research Reported

Programming I is an introductory programming course, designed for students with little or no programming background. Its main goal is to teach the fundamentals of procedural programming. Two programming languages are used, FPL (Taylor, 1982), a graphically represented procedural programming language, and Pascal (Turbo Pascal, Borland International), a traditional, textually represented language. The philosophy and manner of teaching both languages has been reported elsewhere (Taylor,1982; Cunniff, 1988). In terms of the present research, each of these two language implementations used in the course represents a complex environment that must be mastered by the subject student *before* he or she can actually use the software to learn programming. The teaching of programming in this course is initially delayed while many of the students flounder about just learning to navigate the programming language environment of FPL, the first language introduced, then is delayed again, when the second language, Turbo Pascal, is introduced, and they must learn to use its environment. Because we owned all rights for FPL and could, therefore, freely create a special version required by the experiment's design, the FPL environment was chosen as the subject of this research.

Investigating Solutions to the Training (Familiarization) Dilemma

Prior to the experiment reported, an effort to solve the training problem surrounding the FPL system was undertaken, without conducting any research on its efficacy. That effort consisted of creating a minimal manual, which guided the student step by step through a simple scenario of FPL use. Because students did not always read even a *minimal* manual correctly nor follow it exactly, we decided to look at other alternatives to solving this training problem.

In an effort to identify the most efficient alternative, we designed an experiment using both *scenario machine* and *training wheels* interfaces. In two classes, where students were being introduced to programming, we ran an experiment using three different interfaces for training these students to use the actual programming language interface itself. The experiment consisted of three parts:

1. a training session in which two groups used modified interfaces and one group used the full system;
2. a "near transfer" session (within an hour of training) during which all subjects used the full system interface, and
3. a "far transfer" session (approximately one week later) during which all subjects used the full system once again.

The experiment focused on the use of the interface, not on the learning and teaching of programming concepts, and so the subjects were not required to compose programs. In all three phases of the experiment, the subjects were always given complete, correct programs, and required merely to enter and execute these programs.

Our hypothesis was that blocking the consequences of erroneous entry would eliminate some of the frustration and confusion caused in the initial uses of a system, and thereby facilitate learning of a complex system by novices. We argued that facilitation would be reflected in measures of time and total number of keystrokes for the task completion.

Subjects

All subjects were students enrolled in introductory computer courses at Teachers College. Approximately half of the subjects were enrolled in what is considered to be the most introductory level computing course offered, *Computer Applications in Education*. The average student in this course has little or no background in computing of any sort. Most have never used a computer before enrolling in this class. The remainder of the subjects were enrolled in the introductory programming class, *Programming I*. Although Programming I is a more advanced course than Computer Applications, most of its students have little or no background in programming either, and many have never previously used a computer. However, in both classes there are some students who are experienced word processors and some who use computers on a regular basis at work or for completion of school assignments. Consequently, although all of the subjects were students in introductory computing courses the wide range of ability and computer experience impacted the data significantly.

40 subjects participated in the experiment. In this paper we report findings from 21 subjects. Others were excluded either because they did not complete all three parts of the experiment or because the data were not complete.

Method

The subjects were randomly assigned to one of three treatments: **Scenario Machine(TW1)**, **Training Wheels(TW2)**, or **Full System(TW3)**. Each group participated in a **training session**, a **near transfer session** and a **far transfer** session.

All three phases of the experiment took place during class periods. The near transfer treatment was administered within an hour of the training. The far transfer session was administered one to two weeks later. All experiment sessions took place in a teaching lab of IBM-XT computers. Each subject was working at his/her own computer, but the workstations were in fairly close proximity to one another, and so there was some verbal interchange between and among subjects.

During each of the three sessions, the subjects were required to enter and run a simple computer program. The programs were written by the experimenters. Handwritten copies of a complete and correct program were given to the subjects with the directions that they were to enter the program **exactly as it appeared on the sheet**. The three programs were different in context and purpose, but were similar in structure, all using the same subset of programming constructs and commands. All of the structures and commands needed for the near and far transfer programs were present in the program used in the training session. However, the programs for the near and far transfer sessions required the subjects to use some variations of the constructs and syntax of the programming language.

Training Session. At the beginning of the training session, each subject was given:

1. a single page of written instructions indicating how to get started and how to finish,
2. a single page with a summary of all of the FPL system commands (in graphics and text)
3. a booklet with a detailed "Guided Tour for Entering and Running an FPL Program" which contained a program to be entered.

The booklet was a complete scenario; every keystroke was illustrated with screen shots, detailed and explained as it guided the reader through the entry of a simple program. If followed exactly, the user would end with a completed, correct program. See Figure 1 for a reproduction of one page of the "Guided Tour" booklet.

The experiment was designed so that during the training session, each group would use a different representation of the actual FPL system. The **Scenario Machine** group (**TWI**) actually did not use the FPL system at all. They entered the required program using a prepared script that had the **look and feel** of the FPL system, but was actually written with Dan Bricklin's *DEMO II* program[1]. This DEMO was a literal scenario machine to the extent that the user was required to enter the program in the exact sequence set forth by the trainer. No stray keystrokes would be accepted; no deviations were possible. If an "incorrect" key was pressed, the computer beeped and no other response was evident. The **Training Wheels** group (**TW2**) used a modified version of the FPL software. The modifications were designed to prevent the user from getting lost in the system and were made after cataloging and studying common errors made by beginners. All non-essential entries were screened out so that if the user tried to enter a section of the system to which he/she had no introduction, that entry was blocked. Thus, although some exploration and side-tracking was possible (particularly in the online help system), potential fatal errors were avoided. The **Full System** group (**TW3**) used a full version of the FPL software. They had all options available.

All three groups were instructed to follow the scenario laid out in the Guided Tour booklet. The instructions indicated that if they followed the scenes in the booklet exactly, they would successfully complete the task.

Transfer Sessions. The materials used for both the near and far transfer sessions were exactly the same as those used during the training session with the exception that the program to be entered was simply presented on a single sheet of paper rather than explained in the Guided Tour booklet. Each subject was given:

1. a single page of written instructions indicating how to get started and how to finish,
2. a single page with a summary of all of the FPL system commands (in graphics and text), and
3. a sheet with a hand-drawn version of the program to be entered.

Subjects in all three groups used the full FPL system software for both the near and far transfer sessions.

Data Collection

At the start of each of the three experimental sessions (training, near and far transfer), a **TSR** (Terminate-Stay Resident) program was invoked on each user station to capture and record in a file every key pressed during the session. The TSR program[2] uses the system clock so that the exact time of each keystroke is recorded and filed (see Figure 2).

For each subject three data files were collected; one from each session. Each provided a complete record of every keyboard entry. The time-recording feature allowed us to measure not only the total time for each session, but also allows determination of time lost by deviation from the intended scenario, pauses in attention, and time lost by "floundering around" in the software.

Findings

The data were examined for information on the following three statistics for each of the three sessions for each subject:

- successful completion of task requirements,
- total time, and
- total number of keystrokes.

Completion. All subjects in the TW1 group completed all three programs. For both the TW2 and TW3 groups, four of the seven subjects failed to complete one or more of the assigned programs.

Time. Figure 3 shows average total times for each group for each of the three sessions of the experiment. Time for completion of the training session is comparable for all three groups. The other two sessions, however, show a marked difference in the time required for completion. This is particularly true for the TW2 group. The pattern for the TW1 and TW3 groups are similar: training time was lengthy, near transfer times were considerably shorter followed by an increase for far transfer. The times for the TW2 group did not follow this pattern, in fact, there is little difference in time among the three sessions.

Keystrokes. Figure 4 shows average number of keystrokes used by each group to complete the three sessions of the experiment. As expected, the Scenario Machine subjects used the fewest number of keystrokes to complete training. The number of keystrokes used to complete the near transfer task is almost identical for the TW1 and TW3 groups while the TW2 group, on average, used almost 25% more keystrokes. In the completion of the far transfer task, the subjects trained on the Scenario Machine, as expected, used the fewest number of keystrokes.

Discussion

Completion

A program was considered complete if it produced the specified results when executed. The restraint imposed by the Scenario Machine insured that all subjects in the TW1 group completed the training exercise. *All subjects in this group were also successful in completing both the near and far transfer tasks.* .

---

[1] Available from Peter Norton Computing.

[2] We would like to acknowledge the support of Steve Taylor, Teachers College, who wrote the TSR program used.

All subjects in the TW2 and all but one of the TW3 group also completed the required program during the training session. Four of these programs(two from each group), although complete, were not identical to the program specified in the training document. For both groups, TW2 and TW3, the near transfer task was completed by five of the seven subjects. The far transfer task was successfully completed only by three of the TW2 group; all but one of the TW3 subjects were able to complete this task.

A detailed examination of the keystroke data reveals that the difficulties encountered by the TW2 and TW3 subjects in completing the transfer tasks were attributable to two factors: the syntactic requirements of the language interface (use of colon to concatenate in external assignments) and misunderstanding of how to enter required keystrokes (use of <CTRL>, <SHIFT>, and <ENTER> keys). Some of these difficulties may be attributed to the lack of previous keyboarding experience by members of these groups. At the same time, the very success of the TW1 group in avoiding these difficulties, underscores a benefit inherent in Scenario Machine training.

Time

During the training sessions, time to complete the task was reasonably close for all three groups. The Scenario Machine used by the TW1 group required that they follow the manual exactly. Entry of an incorrect keystroke was signaled by a beep and the subject was forced to refer to the manual to determine the correct entry. Analysis of keystroke data for this group indicates a time lapse between keystrokes consistent with this conclusion. Study of the keystroke data collected from the TW2 group indicates that they often choose to bypass the option screens and help menus illustrated in the printed scenario. Of course, this indicates that these subjects were, in fact, learning to use the system; they were not, however, following the prescribed training task.

The TW3 group had the same freedom to skip unneeded help. The data for this group indicate that the savings in time achieved by skipping help options was more than balanced by the time lost when the subject entered parts of the system not explained by the scenario. One way to recover from an erroneous keystroke in FPL is to use what we refer to as the "+-C" combination. In actual use, pushing the "+" key at any prompt in FPL provides a menu offering help or the option to cancel the previous entry. Keystroke analysis indicates time lost floundering in unfamiliar parts of the FPL system followed by a series of efficiently entered recovery steps are evidence for this assessment. Having moved away from the intended scenario, subjects in this group often required the use of the "+-C" combination or intervention by an instructor to get back on track.

Time to complete the near and far transfer tasks were as expected for the TW1 and TW3 groups. The times for the TW2 group were a great deal longer than expected. Close examination revealed a great deal of random keypressing in an attempt to get to a familiar screen display. During the training session, the subjects in TW2 were not required to follow the scenario exactly but were prevented from any major mistakes. In the subsequent transfer tasks, the protection afforded by the modified system was not available and the rapid entry of a number of seemingly random keystrokes, a method that had been somewhat successful during training, led to disastrous results on the full system. Intervention by instructors was often necessary to get the subject back on track. This intervention resulted in the training of the subjects in the use of the "+-C" combination. Keystroke evidence indicates that subjects instructed in the use of this key combination subsequently use it efficiently to recover from less serious errors. This might explain the fact that for the far-transfer task, the average keystrokes for TW2 and TW3 were comparable.

Lengthy gaps between keystrokes in time might indicate that the subject was

- waiting for or receiving intervention from the experimenter,
- reading the "Guided Tour" booklet (during training session),
- discussing the task with other subjects,
- attending to online help or menus.

Although the data were not extensively analyzed for information about intermittent time gaps between keystrokes for this paper, this is an interesting and important area for investigation and should inform us about the behavior of beginning users of software systems.

Keystrokes

In all three phases of the experiment, the TW1 group averaged fewer keystrokes than the other groups. Analysis of the keystroke files indicates that the TW1 subjects tended to make very few errant entries. Data analysis also indicates that subjects in the TW2 and TW3 groups used the "+-C" combination three to four times more often than did subjects in the TW1 group. The use of this key sequence to cancel erroneous entry often resulted in large number of repeated keystrokes thus increasing the overall keystroke average. It is interesting to note that the TW2 group did not use the "+-C" combination as often as the TW3 group during the training session because the software would not permit these subjects to enter options that were as likely to cause difficulty. This may explain the jump in the number of keystrokes for the TW2 group during the near transfer session.

Detailed analysis of keystroke data indicates that keyboarding skills play an important role in the success of subjects in this type of experiment. Slow entry of keystrokes and difficulty in finding individual keys was evident for a majority of the subjects. Data from a number of subjects, however, indicates that these subjects were not familiar with the use of the shift key to enter the shifted version of certain keys. This was evident when the subjects were required to enter the "#" and the "*" symbols. The data reveals a string of 3's and 8's in several instances followed by combinations of these keys with the <CTRL> and <ALT> keys. A similar situation occurs when subjects were required to use the <CTRL> O combination. In this case subjects would hold both keys down for an extended time resulting in a long string of ^O's. This evidence is more prevalent for the TW2 and TW3 groups and can be explained by the fact that the TW1 training interface would not permit the subjects in this group to go on until the proper sequence of keys was entered. Consequently, the TW1 subjects were trained in the required keyboarding skills by the rigid requirements of the training interface.

Conclusions

The findings reported in this paper indicate that training novices to use complex software can be facilitated by the use of an action constraining simulation of the actual software. Working in a restricted environment apparently affords the new user the opportunity to use the application software without being sidetracked and disappointed by errors. In such a controlled learning situation, the novice uses the software without the fear of the "fatal" mistake, thus gaining experience with the computer interface and confidence in her/his ability to use the application software.

Our findings indicate that such a simulation or *Scenario Machine* interface provides more efficient training than either the actual system or a somewhat constraining, special version of that system. A scenario machine can be built for most applications using commercially available software[3].

References

Bernard, P. J. & Young, R. M. (1987). The use of scenarios in human-computer interaction research: Turbocharging the tortise of cumulative science. *Human Factors in Computer Systems and Graphics Interface: Proceedings of CHI+GI'87*, 291-296.

Black, J. B., Carroll, J. M., & McGuigan, S. M. (1987). What kind of minimal instructional manual is the most effective. *Human Factors in Computer Systems and Graphics Interface: Proceedings of CHI+GI'87*, 159-162.

Carroll, J. M. (1982). The adventure of getting to know a computer. *IEEE computer*, **15**:11, 49-58.

Carroll, J. M. (1984). Designing MINIMALIST training material. *Datamation*, **30**:18, 125-136.

Carroll, J. M. & Carrithers, C. (1984A). Training wheels in a user interface. *Communications of the ACM*, **27**, 800-806.

Carroll, J. M. & Carrithers, C. (1984B). Blocking human errors in a training wheels system. *Human Factors*, **26**, 377-389.

Carroll, J. M. & Kay, D. S. (1985). Prompting, feedback and error correction in the design of a scenario machine. *Human Factors in Computer Systems: Proceedings of CHI'85*, 149-153.

Carroll, J. M. & Mack, R. L. (1984). Learning to use a word processsor: By doing, by thinking, and by knowing. In J. C. Thomas & M. Schneider (Eds.), *Human Factors in Computing Systems*, (pp. 27-34). Norwood, NJ: Ablex Publishers.

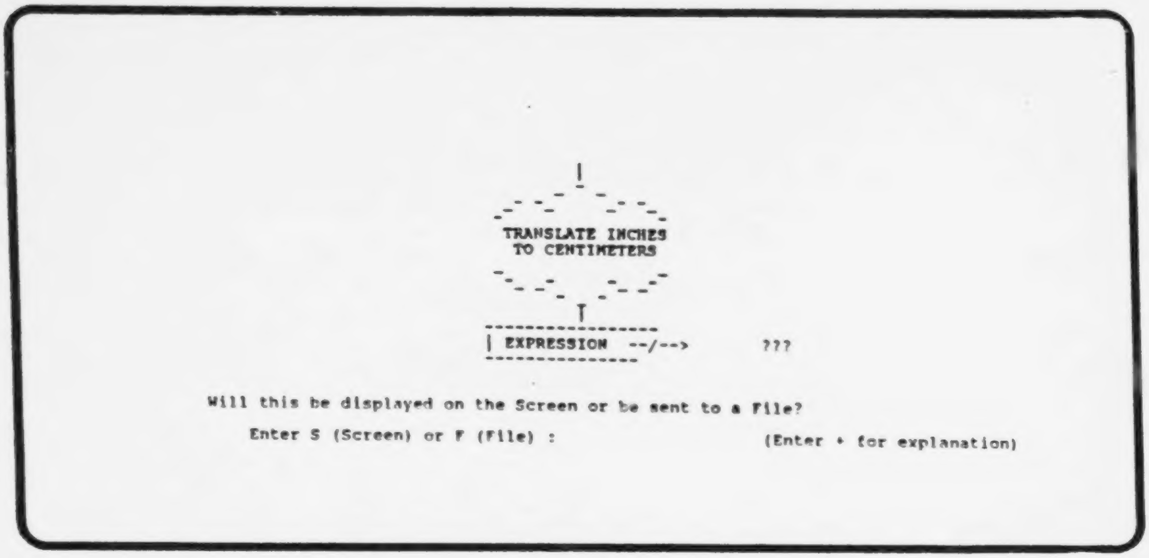
Cunniff, A. (1988). *The Graphical Representation of Programming: A Study of the Comprehension of Novice Programmers*. Ann Arbor, MI: University Microfilms International.

Mayer, R. E. (1980). Elaboration techniques for technical text. *Journal of Education Psychology*, **72**, 770-784.

Shneiderman, B. (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Reading, MA: Addison-Wesley.

Taylor, R. P. (1982). *Programming Primer*. Reading, MA: Addison-Wesley.

Taylor, R. P., Cunniff, N., & Uchiyama, M. (1986). Learning, research and the graphical representation of programming. *Proceedings of the Fall Joint Computing Conference*, 56-63.

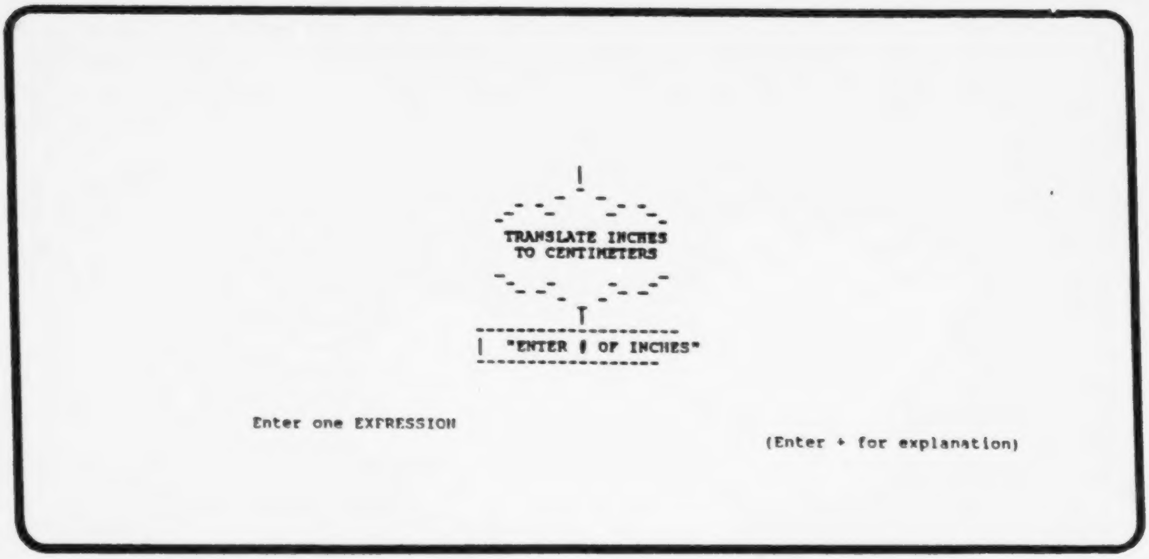


```
                          |
                   _-- --_-_--_ --_
                  TRANSLATE INCHES
                  TO CENTIMETERS
                   -_--_-_   _--_--
                          T
                 -----------------
                 | EXPRESSION  --/-->        ???
                 ---------------

   Will this be displayed on the Screen or be sent to a File?

       Enter S (Screen) or F (File) :                  (Enter + for explanation)
```

Scene 31 : Here another choice was required: was the information to be transmitted to the screen of the computer or to a floppy or other disk file? Since we want it displayed on the screen in this particular program, we entered an S . Do the same thing.



```
                          |
                   _-- --_-_--_ --_
                  TRANSLATE INCHES
                  TO CENTIMETERS
                   -_--_-_   _--_--
                          T
               ---------------------
               |  "ENTER # OF INCHES"
               -------------------

     Enter one EXFRESSION
                                                    (Enter + for explanation)
```

Scene 32 : The actual contents of the external assignment message should be typed now. As we did, type "ENTER # OF INCHES" Don't forget to type the quotes!

FIGURE 1: A page from the "Guided Tour" Booklet

Sample 1:

```
d     17:58:25.41
+     17:59:09.68
<-    17:59:11.22
e     17:59:56.70
<-    17:59:58.13
e     18:00:53.44
<-    18:00:53.88
e     18:01:12.44
<-    18:01:12.55
i     18:01:20.62
n     18:01:20.79
c     18:01:20.90
h     18:01:21.01
<-    18:01:21.34
i     18:01:35.01
<-    18:01:38.36
2     18:02:00.50
<-    18:02:00.61
<-    18:02:24.23
E     18:02:35.38
<-    18:02:35.43
C     18:02:44.99
M     18:02:45.10
```

Sample 2:

```
a     16:59:04.92
<-    16:59:06.18
<-    16:59:07.67
a     16:59:10.80
r     16:59:11.07
e     16:59:11.29
a     16:59:11.51
1     16:59:12.61
^H    16:59:13.33
^H    16:59:13.49
^H    16:59:13.65
^H    16:59:13.82
^H    16:59:13.98
s     16:59:14.26
w     16:59:14.53
^H    16:59:17.11
q     16:59:17.50
u     16:59:17.83
a     16:59:17.88
r     16:59:18.10
e     16:59:18.21
<-    16:59:18.49
```

FIGURE 2: Samples from 2 keystroke data files

Figure 3: Average Time to Competion

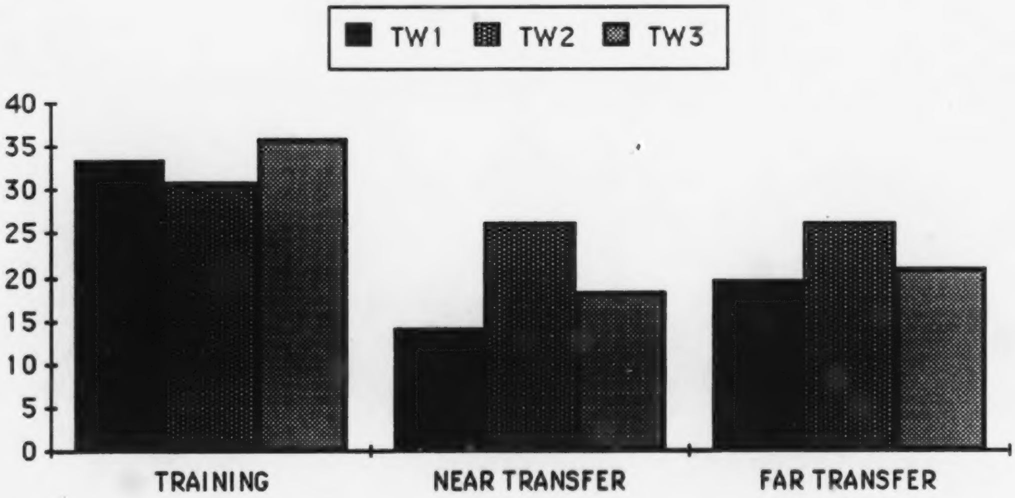
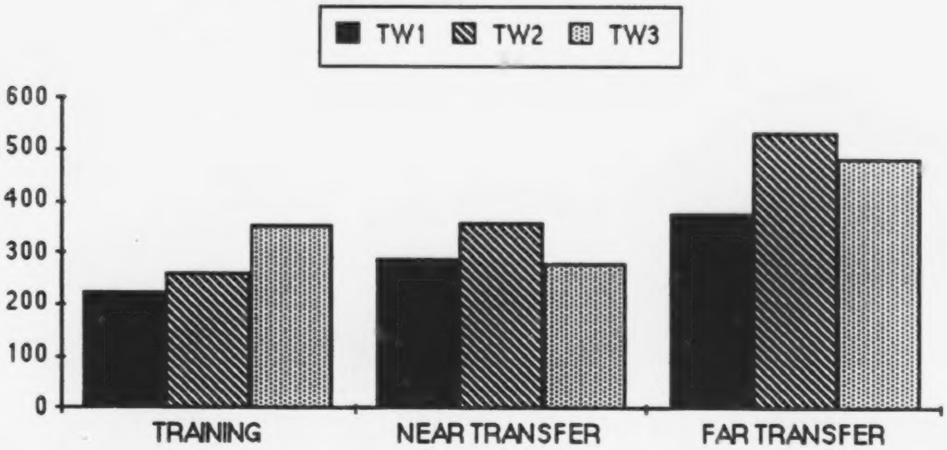


Figure 4: Average Number of Keystrokes



PREVENTING PROGRAM PLAGIARISM: EDUCATIONAL APPROACHES

Marguerite Summers, Chair
Murray State University
Murray, KY

ABSTRACT

Faculty are fully aware that prosecuting cases of suspected plagiarism is unpleasant. This task frequently results in greater problems for the faculty member than the individual accused of plagiarism. The administrative procedures involved are increasingly legalistic in nature. Most faculty feel unprepared to function in that kind of environment.

The panelists will examine alternatives for preventing plagiarism through changing faculty and student attitudes toward plagiarism. Panelists will discuss their experiences with educating faculty and students to their responsibilities in the prevention of plagiarism.

Students frequently have difficulty understanding exactly what kind of assistance is acceptable for programming assignments. Faculty are responsible for educating students to the nature of program plagiarism and clearly specifying the course guidelines.

Administrators have responsibilities in changing both faculty and student attitudes. Administrators must educate faculty to: 1) behavior likely to encourage program plagiarism, 2) university regulations governing program plagiarism, and 3) legal consequences. Administrators also have a responsibility to educate students to the general nature of academic dishonesty and the university regulations related to academic dishonesty, including those specifically related to program plagiarism.

Included in the panel are administrators and teaching faculty. The employers of the panelists represent a broad spectrum of educational institutions.

PANELISTS

William B. Evans
Louisiana State University
Baton Rouge, LA 70803

James J. Fletcher
George Mason University
Fairfax, VA 22030

John T. Gorgone
Bentley College
Waltham, MA 02254

Cindy Hachney
Oklahoma Baptist University
Shawnee, OK 74801

SPONSOR: ACM/SIGCSE

PROJECT HEADLIGHT - "SCHOOL OF THE PRESENT"

Marilyn Schaffer, Paula Hooper, Isaac Jackson
Learning and Epistemology Group
Massachusetts Institute of Technology
E15-314, 20 Ames Street
Cambridge, MA 02139

Leo Biehler, Joanne Ronkin
Project Headlight
Hennigan School, Boston

ABSTRACT

This session is planned to report on recent activities of Project Headlight, a school-based project directed by Seymour Papert and the Learning and Epistemology Group at the Media Lab of the Massachusetts Institute of Technology. The project is located in the Hennigan School, an inner-city Boston public elementary school.

The goal of Project Headlight is to help create a "computer culture" in which students and teachers use computers as easily and comfortably as they use pencils and chalk. Children at the Hennigan School spend approximately an hour each day at the computers writing programs, but it would be wrong to say that they use computers for programming. Rather, students learn programming in the context of working on projects which are intregally related to their curriculum studies.

A major focus has been to create a culture in which there is sufficient competence and comfort with computers for teachers and students to use the general-purpose tool of Logowriter creatively in their own personal styles.

Computers have been used at the School for work that overlaps all the usual subject matters and some others as well, such as an invention workshop based on construction kits in which Lego is interfaced to the computer under Logo control. There is virtually no use of prepackaged software in the school.

The conceptual foundation and rationale of Project Headlight is based on a cluster of ideas formulated by Seymour Papert, as he has worked directly with children and students over the past 20 years. The following ideas will serve as the focus for the present session:

Personal Appropriation

"We do not conceptualize our principal goal as the transmission to teachers or students of technical information. A central idea that has guided our work with children, and which applies equally to adults, is that technical information is easily learned if the affective, social and cultural settings allow it to be imbued with a sense of personal meaning. We use the word 'appropriation' to refer to a process larger than what is generally meant by 'learning.' Appropriating something carries connotations of making it your own, assimilating it to your own personal style, investing it with personal meaning and being comfortable with it." (Papert, 1988)

Constructionism

"This is a term we have invented to refer to a methodology of learning. Constructionism is an extension of the constructivist theory of developmental psychology. According to this theory, learning cannot be reduced to the transmission of knowledge. It represents learners as reconstructing knowledge, as building their own intellectual structures. Constructionism adds to this the idea that the internal construction of a knowledge structure works particularly well in the context of building a meaningful entity in the external world. Such an entity might be built by a child using a construction set, it might be a written project, it might be an organization. In our work with Logo, we have seen how computer programs can serve as constructive projects for learning. In work at the Hennigan school, we have seen how building (as opposed to using) instructional software can be an instructive project for children. Our recent work with teachers has used the same principles. Constructionist learning is not simply something we want teachers to learn about in order to offer it to their students. We want them to learn in the same, constructionist ways." (Papert, 1988).

The presenters will describe projects, based on the above concepts, which have been created by teachers, students, and members of the Learning and Epistemology group.

Another focus of the project is describing the process by which the participants of the project learn to work together. Creating successful learning environments and cultures involves paying careful attention to social interactions. Not only are we concerned about issues of the collaboration among children, we are also currently looking at the role of collaboration among adults in the project. We will describe several recent research projects at Hennigan School that have collaboration as a focus.

SAMPLE COMPUTER-BASED MATH CLASSES USING DIRECT MATHEMATICS MANIPULATION ENVIRONMENTS

James J. Kaput
Department of Mathematics
Southeastern Massachusetts University, and
Educational Technology Center
Harvard University.

ABSTRACT

We will show and reflect on video of both novel software as well as "mock" software simulating new forms of computer-based instruction from several grade K-8 mathematics classes implementable on technology that will be school affordable in the mid-late 1990s. The "direct mathematics manipulation environments" will be of four types.

One will exploit a variety of cybernetic manipulatives, directly manipulable screen objects capable of being used to model mathematical operations and relationships in ways analogous to physical manipulatives, but unlike physical objects, they are actively linked to the more formal representations of mathematics. A second links phenomenologically rich and familiar experience whose quantifiable aspects are extracted and modeled in simulated "real time" using standard mathematical representations. The process is also reversible, so actions on mathematical models can be imposed on such experiences in a direct manner. An example is "Math Cars," complete with accelerator, windshield, and dashboard that can display coordinate graphs as well as other quantitative information (e.g., odometers that scroll tables of distance data) that is generated as one drives in simulation mode. This environment is intended to make the basic ideas of calculus accessible in middle school. A third example shows data manipulation environments where the data appear as information carrying screen objects that move about as one modifies descriptions of the data, as in cross tabulations, set descriptions, etc. A fourth example involves active notecards that describe quantities and relations among them used to model problem situations. Students create and modify notecards. The computer tracks their modifications and variations of given data as they ramify across related notecards.

We will close with reflections on (1) such new forms of software now possible, including the new relationships now possible between everyday phenomenally rich experience and mathematical representation systems, (2) the practical implications for curriculum design and pedagogy, and (3) what needs to be done to make the scenarios offered into practical realities.

IMPLEMENTING COMPUTERS IN SCIENCE EDUCATION

Nancy Bathke
James D. Ellis

ABSTRACT: The Rocky Mountain Apple C.O.R.E. Classroom

Nancy Bathke, Woodmen-Roberts Elementary, 1130 West Woodmen Road, Colorado Springs, CO 80919.

In the Rocky Mountain Apple C.O.R.E. Classroom at Woodmen-Roberts Elementary School, twenty-seven fifth graders are involved in a "school of the future" scenario. Each student has an Apple IIgs computer on his or her desk.

The project (which is modeled after Apple Computer's ACOT project) reflects School District #20's research efforts to understand the impact of computers and technology on the teaching/learning environment. The data from the study (such as time on computer, effectiveness of the computer in various curricular areas, etc.) will help determine the district's direction relating to the most appropriate uses of computers and ultimate student access goals.

In conjunction with BSCS/Project ENLIST MICROS, one of the main site goals is the use of computers in science. The students are performing computer based laboratory experiments, investigating science simulation software, and using the computer as a tool to record and manipulate scientific data. Students also work with science software which incorporates tutorial and drill and practice on science-related skills.

Although the site's primary function is research, curriculum development is a necessary component. Many interesting integrated units of study have evolved.

ABSTRACT: ENLIST Micros II: Implementing Microcomputers in Science Teaching

James D. Ellis, The Biological Sciences Curriculum Study, 830 North Tejon Street, Suite 405, Colorado Springs, CO 80903.

The Biological Sciences Curriculum Study (BSCS) with support from the National Science Foundation (NSF), Colorado Springs Public District 11, Pikes Peak Board of Coopratíve Services, and software publishers is conducting a three-year project (ENLIST MicrosII) to develop a model for implementing educational computing in school science. ENLIST Micros II has as its primary goal to develop and test a model for implementing educational computing in science courses.

During the second year of the project, project staff conducted one two-day workshop and four seminars for 22 lead teachers who were to be group leaders and five two-day workshops and four seminars for 80 K-12 teachers who were new participants in the project. Throughout the year the project staff, group leaders, and new participants worked together to improve the use of microcomputers in science teaching.

The leaders and new participants were experienced teachers with the majority having masters degrees. Most of the leaders had used microcomputers in science teaching prior to the project; more than three fourths of the new participants, however, were nonusers or novices in educational computing. The leaders and new participants gave the workshops and seminars high ratings. By the end of the second year 100 percent of the leaders and 84.6 percent of the new participants were using microcomputers to manage instruction and 92.3 percent of the leaders and 66.7 percent of the new participants indicated that their students were using microcomputers to learn science. The profiles of the leaders and new participants on the *Stages of Concern Questionnaire* changed from one typical of nonusers toward one appropriate for users of an innovation.

THE EVALUATION AND SELECTION OF PROJECTION DEVICES FOR THE MICROCOMPUTER

Doris M. Carey, Chair
Technical Committee on Computers in Education
Computer Society of the IEEE

ABSTRACT

The Technical Committee on Computers in Education, a committee of the Computer Society of the Institute of Electrical and Electronic Engineers (IEEE), seeks to inform educators concerned with the use of technological tools in the educational process. To this end, the present session will address the use of projection devices for the microcomputer.

Specifically, the presenter will address the characteristics of transmissive displays used with regular overhead projectors. A list of criteria for evaluation will be discussed, and several projection devices will be available for demonstration.

The session will cover the uses of transmissive displays for the Apple series of computers, the Macintosh, and IBM©compatible computers. The selection criteria and special features will certainly be applicable to many brands of display unit and microcomputer.

SPONSOR: IEEE-CS

304

ELECTRONIC MUSIC IN THE SCHOOLS

Gordon F. Duckel
Newton South High School
140 Brandeis Road
Newton Center, MA 02159

ABSTRACT

This special session will show how an Electronic Music Studio has been incorporated into the academic atmosphere of Newton South High School.

The unique feature of this program is that it uses the computer as a means to a musical end rather than as an end within itself. It is a project oriented course that uses simple understandings and incorporates them with more complex ideas and understandings. The computer thus becomes merely another cog in an array of synthesizers and modifiers with its ultimate function being that of the master controller.

The session will briefly demonstrate several simple and advanced projects designed to familiarize students with the electronic music studio. It will also show how students can improvise, produce, and record music. A half hour performance of original works by three students from Newton South High School will end the session.

While the ultimate goal of the course is a finished recording, the basic philosophy is to find a way for students to have the freedom to create music and sounds, literally without boundaries. The word creativity cannot be overemphasized as it is what separates this program from many others. The program has attracted many students who would have fallen through the cracks in an otherwise successful music education program. It has opened the way to a better understanding of both music and technology.

COMPUTERS FOR MATH, SCIENCE, AND EQUITY

Jo Sanders, Mary McGinnis
Women's Action Alliance, Inc.
370 Lexington Avenue
New York, NY 10017

ABSTRACT

This panel presentation addresses what happens when a school decides to work with outside experts to increase its female students' computer use.

In the first half of 1989 several middle schools became involved as field test sites in the New York State Computer Equity Project, administered by Women's Action Alliance (New York City) and funded by the National Science Foundation. Because the girls in these schools often declined the computer opportunities available to them, computer specialists and administrators hoped that participating in the project would help them solve the problem.

The school districts of Orchard Park, Sherburne-Earlville, Rochester, and Hempstead, and the teacher centers at South Huntington and Penfield all sent representatives—computer specialists, administrators, and classroom teachers—to a training session at which they learned how to conduct a computer equity program in their schools. On their return, they taught a workshop, "What is Computer Equity?" to their entire middle school faculties, and taught a second workshop, "Computer Equity in Math and Science" to their math and science teachers. For the rest of the term they led a team of faculty members, parents, and students who carried out varied strategies to encourage girls' computer use.

The experience raised many issues for the field test participants. For some, the project was an occasion to think through the philosophy of computer education, while it gave others an unusual opportunity to focus on sex fairness in their school even beyond the computer aspect. Participants found being in on the ground floor of materials development a fascinating experience in unexpected ways, particularly since these materials concerned computer education and sex equity—both subjects that can arouse strong feelings.

The panelists—Alliance Project Director Jo Sanders and Computer Specialist Mary McGinnis, and three representatives of the field test schools—will talk about the field test from their various points of view and will answer questions from the audience.

PRACTICAL IMPLICATIONS OF PROJECT EVALUATIONS

Julia A. Stapleton
Gregg Brownell, Dieter Zirkler
Catherine Alifrangis, Jan Funk

ABSTRACT: How School Districts in New Jersey Use Computer Technology to Monitor Student Proficiency for Assessed Skills

Julia A. Stapleton, Instructional Technology Curriculum Specialist, Division of General Academic Education, New Jersey State Department of Education, CN 500, Trenton, NJ 08625.

Students in New Jersey are assessed annually in reading, mathematics, and language arts to identify pupils who meet established minimum levels of proficiency. To provide maximum instructional leverage, many of the school districts use computer technology to gather data about student proficiency as an ongoing process. The 596 school districts in New Jersey range in size from less than 200 pupils to more than 50,000. This disparity of numbers has resulted in a diversity of computer-based technologies used by individual districts to track or record individual student skill achievement.

In order to provide an overview of computerized instructional management systems and software, a project was organized with categories that included skills software with student record-keeping features, comprehensive curricula systems, networked systems, and database software. As part of the project, research was conducted to support the premise that computer-managed instruction programs facilitate the development and mastery of skill.

This presentation will briefly review the project's findings on software and systems that permit educators to track, record, and report on the development of student skills proficiency. Emphasis will be on how the collected materials are being used in New Jersey's school districts to provide connections between technology and assessment criteria.

ABSTRACT: Confirmation Bias & Logo—Ongoing Research

Gregg Brownell, Department of EDCI, Bowling Green State University, Bowling Green, OH 43403; Dieter Zirkler, Systems Research Laboratories.

An overview of ongoing research on confirmation bias and Logo learning will be presented. Confirmation bias is the tendency to look for confirming information which will prove an hypothesis, rather than for disconfirming information which will disprove an hypothesis. With regard to problem solving, the ability to look at both types of information is important. Two studies were conducted to examine the effects of Logo programming on confirmation bias in 1) elementary school students and 2) college students. While Logo programming did not significantly ($p < .05$) affect confirmation bias, elementary students exhibited an unusually high tendency to select disconfirming vs. confirming information. These results suggest that students in elementary school may be more receptive to learning planning and self-reflection skills than older students. Long-term programming instruction (or other specific instruction in self-reflection) beginning in elementary school may increase older student's problem solving skills. Results of a study to attain a developmental picture of confirmation bias as it exists in students across grades 4, 6, 8, 10, and 12 will also be presented.

ABSTRACT: Use of an Integrated Learning System Lab in an Elementary School and its Effect on the Teachers and Students

Catherine Alifrangis, Fairfax County Public Schools, Office of Research & Evaluation, 7423 Camp Alger Avenue, Falls Church, VA 22042; Jan Funk, Fairfax County Public Schools, Computer Resource Teacher, Alexandria, VA.

The Educational Systems Corporation (ESC) Integrated Learning System consists of mathematics and reading curricula for grades 1-6, a management system, and a testing system. ESC provides continuous, multi-sensory instruction through self-paced lessons, enhanced with color, voice, music, friendly characters, and graphics. This system was implemented in a K, 4-6 school and an evaluation of its effect on the school program and the achievement of the students was conducted. Students used the system for three 20 minute sessions weekly, working at their own levels. The teacher accompanied the students and was expected to monitor and assist the students as necessary.

The evaluation consisted of monthly site visits, analysis of individual student records, interviews with all teachers and others involved with the program, a teacher questionnaire, a student questionnaire, pre- and post-achievement tests, and descriptive information on how the system was used. In addition, the ESC System was analyzed for its conformity to accepted principles of learning theory, i.e., types of feedback used, motivational aspects, and curriculum development. This presentation focuses on the highlights of this evaluation study.

USING COMPUTER SIMULATION TO TEACH ECONOMICS: THE CASE OF PEOPLE EXPRESS

John D. Sterman
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA

ABSTRACT

Our increasingly interconnected and dynamic world challenges teachers to find new ways to communicate principles of complex systems to our students. Economics presents special difficulties because the large scale and long times involved in economic evolution have precluded experimentation and the use of the scientific method. New developments in computer simulation of dynamic economic systems now make it possible for students to take the controls of a company or economic system and pilot it to success—or failure. These simulations are integrated with real case studies, integrating the richness of the case method and the rigor of the scientific method. Just as a flight simulator gives pilots experience in flying new aircraft in rough weather, and economic flight simulator allows students to try different strategies and test hypotheses about economic relationships, without risking the real company or waiting for results. The simulations are implemented on Macintosh computers and use an intuitive graphical interface requiring virtually no training. Most of all, they are fun.

The example presented represents People Express Airlines. From a standing start in 1981, People Express Airlines grew to be a billion dollar company by the beginning of 1986. Its innovative management style and structure were praised as the wave of the future, and companies around the world rushed to imitate them. Yet by September of 1986, People Express was nearly bankrupt, and was sold at the last minute to Texas Air. The People Express Management Flight Simulator gives students the chance to fly the company themselves and learn about corporate growth, competition, human resources, and other principles of economics and management.

The software will be demonstrated and the design and successful use of "flight simulators" for management and economics at the Sloan School of Management at MIT and at other institutions including secondary schools and colleges will be discussed.

SPONSOR: SCS

THE COURSEWARE DEVELOPMENT PROCESS
THE PENN STATE STYLE

Marilynne W. Mathias, Coordinator of Instructional Design
Carol A. Dwyer, Instructional Designer
Barbara A. Polka, Programmer
Computer Based Education Laboratory
Penn State University
University Park, PA.

ABSTRACT

The Computer Based Education Laboratory (CBEL) of Penn State University assists faculty with the use of computers as an instructional tool, including locating, evaluating, and developing software. Producing quality instructional software requires adherence to a systematic process. The presenters will offer suggestions to potential CAI/Simulation developers and computer support personnel regarding tools, design considerations, successful management procedure, and pitfalls. Questions and interaction will be encouraged.

The discussion will include:

1. Organization of Penn State computer services, with particular emphasis on the Center for Academic Computing (CAC).
2. The services offered by the Computer Based Education Laboratory, a unit of the CAC.
3. Types of faculty software development projects, including examples.
4. The software development team, design process, tools, and pitfalls.
5. Benefits of instructional design.

SPONSOR: ACM/SIGUCCS

IN PREPARATION FOR COMPUTER SCIENCE AND ENGINEERING AT COLLEGE

Keith Barker, Chair
University of Connecticut
Storrs, CT

ABSTRACT

There is a growing concern in government, in colleges, and in schools about education in computer science and engineering and their related disciplines. There are issues concerning school curricula, teacher qualifications, interfacing with the colleges, and what is expected of entering college freshmen. There are a wide variety of Computer Science/Computer Engineering programs and the pre-requisite needs of the students are all different.

Ninety-seven percent of all schools in this country have at least one computer and it is estimated that less than 15% of all teachers in the US actually use computers in their teaching. So, not only do we have a problem of educating students, we may also have a problem in educating the teachers.

This panel session will deal with the issues of preparing students for entry to college CS&E programs, addressing curricula, the provision of computing resources, attracting students, particularly women and minorities, and the expectations of the college as the students enter.

PANELISTS

Harry Lewis
Harvard University
Boston, MA

Carmine Cardello
IBM Desktop Software
White Plains, NY

Naomi Amos
Trinity College
Hartford, CT

Doug Willet
The Travelers Foundation
Hartford, CT

SPONSOR: IEEE-CS

LANGUAGE AND SOCIAL STUDIES

Judith Sweeney, Chauncy N. Rucker, Suzanne Biegel
Roy Moxley

ABSTRACT: The Daedalus Project and Prodigy: Electronic Connections Between Education and Business

Judith Sweeney, Chauncy N. Rucker, University of Connecticut, Special Education Technology Lab, U-64, 249 Glenbrook, Storrs, CT 06269-2064; Suzanne Biegel, Prodigy Services Company.

The Daedalus Project is a demonstration project concerned with computer use, writing instruction, and telecommunications. The project, funded by the Connecticut State Department of Education, Bureau of Special Education and Pupil Personnel, and Apple Computer, Inc., is the 1987 *Exceptional Parent* magazine Summer Program of the Year award winner. Funded since August of 1986, the project has sought to demonstrate the effects of computer technology and telecommunications on the lives of students with disabilities.

Divided into two distinct phases, the project has worked with students both at home and at school. In Phase I of the project, 20 bright students with physical disabilities were selected at the Hemlocks Computer Camp in Hebron, Connecticut to take part in the project. After two weeks of intensive instruction with computers, word processors, and telecommunications (for half of them), the students returned home with their complete computer systems. During the following year and a half, these students received writing instruction via either mail or modem along with home and school visits from the project staff. Remarkable results were seen in both the quantity of writing and the personal affective growth as seen in their writing.

Phase II of the project involved moving the project directly into the schools in resource and instructional settings. Students in this phase of the project had a variety of learning problems, and all received instruction in telecommunications. The primary thrust of this phase of the project has been toward using computers and modems at school to improve writing and communication skills. During the school year 1987-88, three classrooms at Kramer Middle School in Willimantic, Connecticut took part in this phase of the project.

Beginning in the Fall of 1988, Mansfield Middle School in Mansfield, Connecticut was also added to the project, and we anticipate adding at least one more school during the year. In addition to the writing and telecommunications curricula and the electronic bulletin board at the University, these classrooms will also be using the Prodigy Information Service and a specialized non-profit, private bulletin board set up for us on the service. Prodigy Services has joined with the project to explore these kinds of educational bulletin boards. We believe their extremely user friendly, graphics based service will be a major contribution to educational telecommunications instruction.

Our presentation will focus on the highlights of the Daedalus Project during the last 2 1/2 years, especially in terms of telecommunications. Copies of our curricula material [*Daedalus Writing Curriculum II, Telecommunication: Talking to the World with Computers* (student workbook)] will be covered along with some data and results from the project. We will also be highlighting our connection with Prodigy and the advantages it offers with private bulletin boards and up-to-date news, weather, and educational features.

ABSTRACT: Logo Language Experience and Spelling Development

Roy Moxley, West Virginia University, 604 Allen Hall, Morgantown, WV 26506-6122.

This presentation demonstrates and reports the results of using a program that incorporates the features of a language experience approach to writing and reading for pre-school, kindergarten, and first-grade children. The program, Logo Language Experience, is written in Terrapin Logo for the Apple IIe/IIc. In general, children steadily made progress in developing standard spelling and in writing their stories without the need for dictation. However, unlike the findings of previous research that children go through a stage of invented spelling based almost exclusively on the way that letters sound in words, these "computer" children showed much less dependence on a phonological strategy. They regularly invented spellings based on how letters looked in words as well as on how they sounded. The fact that these computer children used visual strategies from early on is considered of some importance since it is generally recognized that all good spellers must eventually adopt a visual strategy. An argument is made that putting language experience activities on the microcomputer makes it easier for children to write their own stories earlier. This program is not available commercially; however, a free copy of this program and a guide will be sent to those who send a blank disk to Roy Moxley, 604 Allen Hall, West Virginia University, Morgantown, WV 26506-6122.

PRODUCING AND USING SPEECH-BASED SOFTWARE

Lynne Anderson-Inman, Chair
University of Oregon
Eugene, OR

ABSTRACT

One of the most promising advances in computer-based technology for education is the increasing availability of quality software with speeech output. This panel provides an overview of current information and issues related to the production and use of speech-based software. Panelists bring a wide range of experiences producing and/or using software with speech output and will share diverse views concerning appropriate applications and critical issues related to this emerging technology.

PANELISTS

Mary Cron
Vice President, Product Development
First Byte
Long Beach, CA

Bill Adler
Vice President
Street Electronics
Carpenteria, CA

Micheal Hillinger
Executinve Editor
Houghton Mifflin
Hanover, NH

Richard Olson
Professor of Psychology
University of Colorado
Boulder, CO

STATISTICS AND GRAPHING

Richard A. Alo'
Robert J. Beichner

ABSTRACT: A Networked Statistics/Computer Science Computer Classroom

Richard A. Alo', Carol Jones, Department of Applied Mathematical Sciences, University of Houston-Downtown, Houston, TX 77002.

This National Science Foundation funded project supports instrumentation in the Applied Mathematical Sciences Department of an open-admissions college. A classroom is equipped with twenty-five networked IBM-Personal System II Model 50 Microcomputers and a large scale projection screen for demonstration purposes. The equipment is used in two junior level statistics courses which are required of business majors. Students encounter daily assigned classroom problems dealing with decision-making situations which require statistical analysis and interpretation of computer output. Solutions are worked out on computers (two students per machine), as in a lab. Mathematically-shy business students will have a greater appreciation of statistical concepts and their applications to real world problems, more experience in dealing with the popular IBM machinery, and increased interest in pursuing optional senior level math courses in the area of the decision sciences. Business students, who represent fifty-eight percent of UH-Downtown's 1986 graduating class, receive enhanced training in quantitative methods. Ultimately, other faculty will be encouraged to use this instructional lab for courses in computer science, numerical methods and calculus.

ABSTRACT: The VideoGraph Project

Robert J. Beichner, Center for Learning and Technology, SUNY at Buffalo, 217 Christopher Baldy Hall, Buffalo, NY 14260.

The VideoGraph project utilizes microcomputers in high school and college level physics labs. Specifically, we have students videotape "large-scale" motion and send the images into a computer for kinematics data collection and graph production. The hope is that by permitting study of real world situations (cars accelerating in parking lots, a person shooting a basketball foul shot, etc.), students will find it easier to make the connection between the often idealized textbook concepts and physics in the world around them.

In our study of 270 students, we compared learning in computerized and traditional (strobe photograph) lab settings. We also examined the effect of actually seeing the motion to working only with photographs or computer images. Our findings will be discussed at the conference.

The project was supported by a grant from the National Science Foundation.

MULTIMEDIA DESIGN

Robert Mohl
Margo Nanny
Apple Multimedia Group
3220 Sacramento Street
San Francisco, CA 94115

ABSTRACT

Learning (and teaching) new things is often challenging. Some concepts are hard because they are hard. Some concepts are hard primarily because they are hard to verbalize and hard to visualize. Many of the latter can learned and taught more easily and more profoundly with the appropriate technology. Apple's Multimedia Lab is producing an experimental project which demonstrates how.

It is called the Visual Almanac and consists of a videodisc, HyperCard software/tools and a teacher's guide. Subject areas touched on vary widely across disciplines and grade levels. Topics include Playground Physics, Experiential Astronomy, Biomes, A Day Around the World, Historical Time Atlas, etc. In additional we have developed tools to allow teachers and developers to do original research and create their own presentations using the 7000 multimedia objects (raw images plus data) in the Visual Almanac. Those familiar with HyperCard will be encouraged to create their own custom activities with as much complexity as they desire.

The Apple Multimedia Lab is a combination of workshop and playground. There is no way to get all the great teachers and educational developers into the Lab to create wonderful new software, so we decided to export the lab. The Visual Almanac is our attempt to do that. We believe that everyone should have access to being multimedia composers. We are sure that given the right tools, a reasonably intelligent and motivated person can, in fairly short order, gain fluency in computer-centered multimedia production. Our goal was to make a set of building blocks to make that fluency possible. We are taking a first step.

TEACHER-LINK: AN ELECTRONIC CULTURE

Judi Harris
Curry School of Education
University of Virginia
292 Ruffner Hall
405 Emmett Street
Charlottesville, VA 22903

Abstract

Information comes in many forms. The exchange of "how-to" information is one of the bases upon which student teaching is built. The other part of the foundation, of course, is direct teaching experience, for which there is no substitute. But if student teachers were able to easily access their supervisors, professors, comrades, and cooperating teachers from their school placements or their homes at any time of the day or night, we at the University of Virginia believe that the quality and quantity of their learning may be improved.

Teacher-LINK is a joint venture between the Curry School of Education and IBM's ACIS division. IBM has generously supplied approximately 1 million dollars' worth of hardware and software so that they can learn, along with us, whether easy access to a telecommunications network of all participants in the student teaching experience can facilitate information exchange, and, concomitantly, learning. The purpose of this presentation is to describe the goals, structure, and interim progress of this effort, with an emphasis on practical aspects of facilitating diffusion of such innovations in public schools and schools of education.

Teacher-LINK: An Electronic Culture

I would like to tell you about a computer-facilitated network of people; namely, student teachers, cooperating teachers, supervisors, and University faculty.

Do you remember your student teaching experience? I remember mine. In the eight weeks that I taught sixth grade at an urban elementary school, I saw my student teaching supervisor twice. My cooperating teacher stayed in the classroom with me for the first two weeks, then left to receive training for the assistant principalship position to which she aspired. One of my professors made it a point to visit each of the undergraduates doing practica and observe a science lesson. But since there were so many of us, she was able to visit each only once. I had no convenient way to contact my supervisor or professors with questions, concerns, or requests, since they, too, worked hard, and were often not in their offices during my lunch break. I had a choice: play pink slip telephone tag, or try to figure it out on my own. I unfortunately didn't have time to play games. My supportive copatriots, whom I had seen every day for seminar during the first six weeks of the semester, were scattered all over the city, so even moral support was not readily available. And yet, I considered my internship experience to be a good one; I had heard stories of much worse.

As you can probably remember from your own experience, this is not an unusual scenario. And, while many teachers-in-training have very positive pre-professional experiences while student teaching, we at the Curry School of Education feel that practica can be made even better if communications links between the University and the local schools are bolstered. One way that we are exploring this possibility is through the establishment of a computer telecommunications network called Teacher-LINK.

Information comes in many forms. The exchange of "how-to" information is one of the bases upon which student teaching is built. The other part of the foundation, of course, is direct teaching experience, for which there is no substitute. But if student teachers were able to easily access their supervisors, professors, comrades, and cooperating teachers from their school placements or their homes at any time of the day or night, without bothering them with a late-night phone call, we believe that the quality and quantity of their learning may be improved.

Teacher-LINK is a joint venture between the Curry School of Education and IBM's ACIS division. IBM has generously supplied approximately 1 million dollars worth of hardware and software so that they can learn, along with us, whether easy access to a telecommunications network of all participants in the student teaching experience can facilitate information exchange, and, concomitantly, learning.

To date, there are approximately 120 teachers in the city of Charlottesville and Albemarle County, Virginia, that have formed and are maintaining the network. Twenty percent teach teachers in the school of education, 40 percent teach teachers in the community schools, and 40% are learning to be teachers. More than half of the networked teachers have learned to use computers and electronic mail within the last four months.

Each supervising teacher that works in the public schools and is participating in the project is loaned an IBM convertible, or portable, computer, a color monitor, and a Proprinter. Student teachers are given portable computers and monochrome monitors to use. Telephone lines into the teachers' classrooms are provided by the participating school districts, with some assistance from Centel. Participating faculty members that do not have access to a computer with telecommications capabilities are given desktop machines and/or modems to use in their offices. Supervisors have access to both shared desktop computers and personal portable computers. All participants are given telecommunications software (ProComm), word processing software (DisplayWrite Assistant), and their choices from a wide range of tool and instructional software. IBM has supplied more than 100 different titles from which the teachers can select.

The focal application of the project, though, is the use of electronic mail. For this, IBM has given the Curry School a 4361 mainframe computer running VM/CMS, which we have nicknamed "Jabez," after Jabez Monroe Curry, of course. Participants can connect to Jabez from their classrooms, their homes, their offices--literally anywhere that there is a telephone line, and at any time that they please. LINKers can write messages to anyone that has an account on the system, then ship them off with two keystrokes to wait in the incoming electronic mailbox of the addressee. When the addressee next logs on, they read the messages waiting for them, respond, write new messages, and the communication continues. All that is needed to send notes is knowledge of the person's nickname on the system, which can sometimes cause some embarrassment. Soon after training a group of student interns, I received this note in my incoming mail:

> Dear Judi, No one calls me James except when I've done something terrible. I haven't done anything, yet. Please, please, please change my nickname to Jamie. I'll sleep much easier.

Jabez is also linked to an inter-University, international network called BitNet, so that our Teacher-LINK participants can just as easily write notes to educators here in Virginia as they can communicate with professors in California and student teachers in British Columbia.

But you have not yet read one of the more powerful potentials of this teacher network. Participants can write documents with a word processor, save them to disk, then mail them over the telephone lines to anyone else in the system. This is called uploading. The recipient can then save the document to their data diskette, load it into their word processor, append, edit, and comment upon it, and mail it back. This can revolutionize the ways in which student teachers submit lesson plans and receive feedback from cooperating teachers and supervisors. While tired teachers and interns may not want to stay for several hours after school giving and responding to feedback, they may find it easier to look at lesson plans, evaluations, and project ideas in the comfort of their own homes. Then, if the intern prefers to work until midnight, but the cooperating teacher chooses to go to bed early and review the uploaded unit outline at 7 am, neither wakes the other, and no time is lost in leaving telephone messages. One of the student interns expressed it well in practical terms in a message to me:

> Judi, just wanted to let you know that I just sent my first ASCII files over the wires and everything seemed to go A-OK. My teacher has a little baby and always has to rush home to relieve the babysitter. We have a hard time finding enough time to talk at school, so I just sent her a couple of lesson plans I would like her to review. I think this will be very helpful!

One of the most important aspects of the Teacher-LINK project is the training that Curry School staff provides for all participants. Just as a surgeon is often the most difficult hospital patient, a teacher is usually a vocal, no-nonsense-tolerated pupil. Her underlying question remains the same: "How will this help me to help my students?" Instruction for teachers must be well-structured, time-efficient, reasonably paced, respectful of individual intelligence, experience, and learning differences, and, above all, practical. With these ideas in mind, training for Teacher-LINK participants was crafted and is being implemented both inside the University and at more than a dozen public school sites.

Three or four two- or three-hour hands-on workshops in Curry's networked IBM classroom are given to introduce participants to use of their equipment, electronic mail, word processing, and file transfers. Optional workshops are conducted for review of instructional and personal productivity software. After initial instruction, technical and training support is maintained by site visits to the community schools, office visits to University participants, use of the mail system itself, and even an occasional application of a more antique technology: the telephone call.

Participants seem to enjoy the training sessions. As one electronically exclaimed,

> Wow! This is proof that you CAN teach an olddog new tricks! Very exciting! Hello. I'm very excited, and intimidated; but,all will be well.
>
> Thank you for sending my first piece of electronic mail. If you are reading this message, then you and I have both done something right.

And the fun continues as they connect from home, school, and office.

> Hi there! This is your local enfant terrible, playing with the new toy.....

Also:

> Judi, don't keel over in a dead (as opposed to non-dead) faint. I have finally logged onto my beloved computer.

And:

> Hi ho! My cat Bill is sitting here with me helping me use the computer. No wonder I had so much trouble tonight getting on. Bill is computerilliterate. Not only that, he is a

> practicedexpert at finding exactly the page of directionsthat I am reading and positioning himselfdirectly on top of that for a nap.

Teacher-LINK participants are beginning to explore some rather original uses of the telecommunications network.

> My students are ready to start writing to their [electronic] pen pals, and we will begin as soon as we get a list of [the third-graders'] names from Betsi. I think it would be great ifyou could plan to be here [at the high school] the first time we try to send out letters on the computers.

> I mean, suppose there were an e-mail address forSecondary English...If I'm a (computer-dependent)ST and I've done my lesson plans/handouts on [aword processor], anyway, and the lesson went reallywell, wouldn't I just love to be immortalized...by sending my brilliant lesson plan to a file whereit could be read by others?

A middle school social studies teacher and intern have set up two electronic mail files, called OLDWORLD and NEWWORLD to assist their students' explorations of turn-of-the-century American immigration. The students are in pairs from a particular country, one writing as if they are still in "the old country," and the other as if they are reporting on their experiences of the "streets lined with gold" in America. When the simulation is over, the students will find out who in the class has been writing to them.

Several elementary school teachers are developing fictitious personalities for installation on the mail system. One is a language bug, whose messages have to be corrected before they can be answered by giggling third-graders. Another is someone who signed the Constitution and knew Thomas Jefferson, and is all too delighted to answer fifth-grade queries about what things were like in the colonies. At the suggestion of a Curry School children's literature professor, electronic mail accounts were created for class novel protagonists, so that students can communicate with them, encouraging deeper inquiry into their personalities as revealed through the book. And graduate students who volunteer to electrically impersonate the characters seem to welcome the excuse to revisit fiction for children.

A supervisor used the network for some informal research.

> Is there a way to set up a number of note logs so that I could establish, say, a separate log for notes from each of my student teachers? Iask because I'm thinking of doing my comps paper on supervising, and such a filing method would make things much easier, methinks.

The network can make rather mundane chores more pleasant,
also.

> Traci and I have been using the email. She communicated a set of test grades that she had done on Friday--I didn't have to wait until Monday to learn the results.

Perhaps one of the most important uses of our teacher network is simply for moral support. There even seems to be some evidence that the network encourages relationships that may not have manifested without the telecommunications link. The example that comes to mind is a romance that developed between two student teachers that work long hours in two different schools. That was one application that I can safely say we didn't anticipate. But my all-time favorite electronic mail message came from a kindergarten teacher in the fall of 1987:

> Dear Judi, Thanks for the Super Sleuth title; I needed a little smile at the end of my exhausting day. I have a child who bangs her head on the floor if she doesn't get her way, another child who throws himself on the floor and yells and screams and kicks his feet in a circle, a third child who cannot express her simplest needs.....and so the list goes. I think I have a child who can read the New York Times. I am supposed to teach these kids about the Constitution. Somewhere in all of this you just have to have a good sense of humor or you will go nuts. When I asked the class about The Constitution, one child told me it is 'when you can't go to the bathroom. Time for recess.

Sociologist Daniel Bell has argued that since the Second World War, American society has been undergoing a process of transformation from an industrially-based to an information-based economy. Information and knowledge are the crucial variables in post-industrial society. Anthropologists have long argued that communication is the basis of culture. Rather than property and production, knowledge and how it is communicated are the crucial issues in the definition of culture. (Provenzo, 1986) Now, thanks to the widespread use of microcomputers, there is a new and powerful way to augment traditional communications networks.

As Glen Bull, one of the project directors for Teacher-LINK, has written,

> ....we have found that this project is more about social systems than about hardware. The central question of Teacher-LINK now appears to be one of how to build a new culture. (1988, p. 19)

References

Bull, G. L., et. al. (1988). Teacher-LINK Interim Report. Charlottesville, VA: University of Virginia, Department of Educational Studies.

Harris, J. B. (1988). [Electronic mail messages received via University of Virginia's local area network]. Unpublished raw data.

Provenzo, E. F. (1986). Beyond the Gutenberg galaxy. New York: Teachers College Press.

NEW HAMPSHIRE'S GOVERNOR'S INITIATIVE FOR EXCELLENCE IN EDUCATION

Michael P. Goldsmith, Arthur Hammon, Larry Bickford
Cynthia A..Dunlap, Marybeth Stock

ABSTRACT: New Hampshire Governor's Initiative for Excellence in Education: Computers For Teachers

Michael P. Goldsmith, Concord School District,16 Rumford Street, Concord, NH 03301; Arthur Hammon, McIntyre School; Larry Bickford, Hopkinton High School.

The Computer's For Teachers Project is one of the components of Governor John Sununu's Initiatives For Excellence In Education. The Computers For Teachers component used $2.8 million state funds and additional local monies to provide computers, peripherals, software, training, and support to more than 25% of the State's regular classroom teachers. The overall goal was to allow teachers to learn to use computers as productivity tools to streamline reporting and classroom management burdens.

The Hopkinton and Henniker school districts ran a joint project which placed computer systems and software in the hands of 24 teachers in grades 4 through 8. The teachers were assigned Macintosh computers and software. Teacher applications include lesson plans, tests, classroom materials and visuals, grading, etc.

The White Mountain Regional school district has involved all the teachers in the district. Teachers received Apple IIe systems and software. Teacher applications include student progress reports, grading, lesson plans, etc. All teachers send progress reports, prepared on the computer, home to parents every three weeks. This has greatly increased parental involvement in the the student's education in the district.

ABSTRACT: New Hampshire Videodisc Development Project: The Granite State Sampler

Cynthia A. Dunlap, Marybeth Stock, Southeastern Regional Education Service Center (SERESC, Inc.), 11 Peabody Road, Derry, NH 03038.

The Granite State Sampler is a two-sided laser videodisc, designed to enhance the New Hampshire History and Government curriculum: it incorporates 12,000 still images and over 50 minutes of archival motion footage for use by the classroom teacher. Monies from the Governor's Initiatives for Excellence in Education were granted to SERESC to produce this videodisc which will be distributed in Spring 1989. Each public school in the state of New Hampshire will receive a free copy of the disc as part of the grant.

Conceived in the winter of 1987 by a group of media specialists, The Granite State Sampler is an innovative visual resource library consisting of thousands of New Hampshire-related images. Initially designed as an archival resource to preserve educational materials (filmstrips, slideshows) that are no longer produced, the concept of the Granite State Sampler grew to include photographs, artwork, documents, and other images selected from diverse statewide sources. It may be considered one of the most versatile, comprehensive archival resources of New Hampshire historic images and film to date.

What was known initially as the New Hampshire History Videodisc Project began in February, 1988 when two committees of educators, historians, librarians and media specialists met to create a New Hampshire history topic outline. Using this outline of broad subject areas, resources and leads were developed. Throughout the summer of 1988, photographs, a/v presentations, and films were previewed and selected for inclusion on the disc. Sources for these images include private and state photographic collections, university archives, and historical agencies and museums throughout the state.

These visual collections cover topics that include state geography; natural history; archeology; ecological concerns in the state; colonial, revolutionary, 19th century, and Depression periods; constitution and government; economic issues; recreation and tourism; interesting people; and special collections unique to our state.

SERESC, the Southeastern Regional Education Service Center, Inc., is a private, non-profit educational resource collaborative of 13 school districts that provides a variety of services to its members, including special education services, education technologies training, and software libraries. For more information: (603) 432-9442.

PREPARING COMPUTER-USING EDUCATORS IN 1989

Margaret L. Niess
Associate Professor
Dept. of Science, Mathematics & Computer Science Education
Oregon State University
Corvallis, Oregon 97331
(503) 754 - 4031

Abstract

Since 1980 teacher education programs have increasing recognized the need to prepare *computer-using* educators. Yet, with the dynamic nature of computing technology, both inservice and preservice teacher education programs must constantly be reevaluated and updated. This project considers the question, "What does it mean to prepare computer-using educators in 1989?"

The Northwest Council for Computer Education (NCCE) funded this project to develop an updated set of competencies which might be used as guidelines in reassessing teacher education programs. The project is a follow-up to the original set of competencies developed in 1983 by a similar grant. Competencies in the updated set are developed for nine categories of educators:

1. all elementary teachers;
2. all middle school teachers;
3. all high school teachers;
4. elementary computer teachers;
5. middle school computer teachers;
6. high school computer teachers;
7. elementary district/building computer coordinators;
8. middle school district/building computer coordinators;
9. high school district/building computer coordinators.

For each of the nine categories, at least 20 specialists (including classroom teachers, curriculum coordinators, and computer coordinators at all levels) identified, refined and evaluated sets of competencies in a four-round Delphi approach.

Questions

Since 1980 the impact of computer technology in society has markedly increased. For example, shorthand has practically vanished from the office of 1989; computers, electronic typewriters and dictating machines have replaced the need for an expert in shorthand. In fact practically every aspect of American business has been affected by computing technology. What about education? Has computing changed how America educates its citizens? What about the teachers? Are they prepared to incorporate computing into the classroom?

Businesses have had to invest in massive retraining projects as a result of the impact of computing technology. They have had to provide time and training for employees to develop the skills necessary to operate the new technology in a productive manner. What about education? In the early 1980s schools in America made huge hardware and software purchases; and, yes, massive inservice and preservice teacher training accompanied the purchases to help the teachers become computer-using educators. With this retraining teachers were expected to teach both with and about the computer, thus preparing computer literate citizens. Yet, the recent National Assessment of Educational Progress (NAEP) project titled "Computer Competence: The First National Assessment" indicates that schools have only dented the ignorance most students and teachers have of computers. The study of over 20,000 students in the third, seventh and eleventh grades found that "schools have failed to use computers effectively and teachers are trained inadequately." Marc Tucker, President of the National Center on Education and the Economy, commented that, "The difference is between knowing a little about the computer, and being able to use it effectively. Few American school children can do the latter."

The problem cannot be resolved by additional investment in hardware and software. The problem requires an investment in the **education of teachers.** During the past ten years, teachers at all levels (elementary, middle and high school) have been expected to provide a computer literacy program for their students. They have been expected to assist students in using the computer as a learning tool. Their preparation for teaching with and about computer technology has largely been from their ability to teach themselves after a few hours of inservice training. Typically, their own experience with computers and the use of computers in education is only "hours" old. They are learning how to learn with the computer but they lack the experience of learning with the computer over twenty years of education. Yet, they are expected to integrate computer methods among a variety of instructional methods. And, of even more concern, they have been expected to "create" computer literacy course work as well as teach it.

The results of this approach are certainly represented in the NAEP assessment. The approach has led to a curriculum that is highly dependent upon the teacher's education. In 1989 are the students allowed, even encouraged, to use their understanding about the computing tools in learning in mathematics, English, history, ...? Do teachers understand the concept of using the computer as a learning tool? Have they been prepared to be a computer-using educator? And, the bottom line for teacher preparation is a consideration of this question: What must a computer-using educator be able to do in 1989? Have the requirements changed in the past five years?

Background

In 1983 the Northwest Council for Computer Education (NCCE) funded a project to develop a set of competencies which might be used as guidelines in developing teacher training programs to prepare teachers in the use of computers in education. This project provided recommendations for the following categories:

1. General: Certification for all teachers.
2. Specific/Elementary: Teachers capable of teaching about computers and computing at the elementary school level.
3. Specific/Secondary: Teachers capable of teaching about computers and computing at the secondary school level.
4. Specific/District/School Computer Curriculum Coordinators: Coordinators capable of planning, developing, organizing, and directing computer science programs for districts/schools.

The guideline competencies were published nationally, and many teacher training programs were able to use the competencies to discuss and develop courses/programs to prepare teachers in the use of computers in education.

However, computing technology and its influence on education has significantly changed since the development of the original guideline competencies. Widespread and everyday use of application packages including word processing (with spelling and grammar checkers), data base systems, spreadsheets, and graphics packages have altered not only the manner in which teachers and school administrators do their job, but have also altered the nature of the curriculum. Nationwide the curriculum is emphasizing learning to use the applications of computer science rather than learning computer programming. In 1983 the topic of telecommunications was not even considered in the development of the competencies. Yet, today, many schools provide some instruction (and hands on experience) in this area. But, a more serious question for an today's education is, How will telecommunications change the way Americans communicate? Should education respond to the need to be able to communicate in a different manner? Where do the teachers learn about telecommunications and its impact on society? Should they have some experience with telecommunications in their preservice education?

The changes that are taking place are likely a reflection of the nature of the marketplace and integration of the technology into everyday lives. Clearly, societal forces have escalated in pressure for citizens who can make good use of computer based aids; to meet this pressure, students must grow up using computers and must receive appropriate training in their uses. In the past five years, more teachers in all disciplines and at all levels, have been required to become involved in computer education.

The International Council for Computer Education (ICCE) Teacher Education Committee proposed the following:

> When making recommendations on teacher education for computer education, a minimum of three audiences must be considered: 1) all teachers, 2) computer science teacher, and 3) district/building computer coordinators. In addition, any set of recommendations must consider the special needs of the elementary, middle and high schools and have a unifying theme emphasizing the concept that the computer is a problem solving tool for use throughout the curriculum
> (Jongejan, et al, 1987).

Furthermore, this committee suggested that all future teachers should be learning about 1) the computer as a system; 2) computer applications in education; and 3) the computer as a tool for personal use.

Teacher preparation for computer education typically recognizes many of the competencies recommended in the original project. Many teacher preparation programs require that the preservice teacher demonstrate competence in using computers in an educational situation. However, in light of the NAEP assessment and the changes in the past five years, it is time to reassess those requirements. More importantly, an updated set of competencies might encourage:

1. teacher preparation programs to reevaluate and update their computer education programs;
2. teachers to reevaluate their own competencies and update their own preparation; and
3. educators to reassess the progress that has been made in computer education and the preparation of teachers to meet the changes.

Project Design

The goal of the project is to develop a set of competencies which might be used as guidelines in updating/developing teacher education programs to prepare computer-using educators. The categories of the competencies, developed through a modified Delphi technique, include nine categories:

	ALL TEACHERS	COMPUTER SCIENCE TEACHERS	DISTRICT/ BUILDING COMPUTER COORDINATORS
ELEMENTARY SCHOOL	1	4	7
MIDDLE SCHOOL	2	5	8
HIGH SCHOOL	3	6	9

Approximately 20 specialists developed and evaluated the competencies for each category. The specialists were selected from Oregon and Washington including classroom teachers at each level, computer coordinators at each level, and curriculum coordinators at each level. Size of schools, geographical region, and types (districts versus schools) of institutions were also considered in the selection of the specialists. As a result, the specialists represented the variety of educational situations throughout the Northwest.

A modified Delphi technique approach was used to develop the set of competency guidelines. The first round proposal sent to the specialists included the set of competencies identified in the 1983 NCCE competency project and suggestions from the ICCE Teacher Education Committee. The specialists were asked to prepare a list of competencies that they believed ought to be included in the identified cell. They were asked to carefully consider their own teaching situations and the directions they see in their school. They were allowed to select competencies from the previous set, to edit some of those competencies, and to identify new competencies. The responses from this round indicated a significant shift from the original set of competencies. Use of the computer as both a personal and professional tool for all teachers at all levels was clearly stated.

The second round combined all the responses from the first round. Specialists were asked to judge the sets using the scale of 1) Strongly Agree, 2) Somewhat Agree, 3) Neither Agree Nor Disagree, 4) Somewhat Disagree, or 5) Strongly Disagree. The specialists were also asked to add any competencies they felt had been overlooked and to clarify any competency they felt was confusing, unclear or misplaced.

The mean rank for each of the competencies within each category was computed for the next draft. The competencies were ordered according to rank within each category. The editorial changes and the additional competencies from the second round were included in this draft. For the third round, the specialists were asked to indicate their judgement using the same scale. They were also allowed to edit and add competencies.

The final draft of the competencies within each category was identified using the mean ranks from round three. All categories were placed in a single document and sent to the entire group of specialists for a final ranking using the same scale.

Mean rank scores were calculated for each item in a category. The final set of competencies recommended by this project for use as guidelines were ordered and presented including the mean ratings.

The Competencies

Since this paper for the *Proceedings* was prepared prior to the completion of the project, the final set of guideline competencies will be presented and discussed at the conference. Additional copies may be obtained by writing the author.

References

1. Associate Press Release (1988).
2. Jongejan,T, D. Moursund, M. Niess (1987). ICCE teacher education committee: some thoughts on preparation of regular classroom teachers, computer science teachers, and computer coordinators. *SIG Bulletin*, 3:4:38-42.
3. Moore, M. L. (1984). Preparing computer-using educators. *The Computing Teacher*, October: 48-52.
4. Moore, M. L. (1984). Preparing teachers to teach about computers and computing. *Proceedings of the NECC '84*. 286-290.

GIVING TEACHERS THE TIME TO TEACH: THE CLASSROOM MANAGEMENT SYSTEM IN THE PUBLIC SCHOOLS IN SOUTH CAROLINA

Gary West
CMS Coordinator
Greenwood School District 50
P.O. Box 248
Greenwood, South Carolina 29648

Abstract

This paper presents an overview of South Carolina's Classroom Management System, a pilot program to develop teacher-level management tools that will computerize and automate the assessement and diagnosis of student needs, the selection and application of instructional prescriptions, and the recording and reporting of the results of those processes and procedures.

CMS is traced from the the State's Education Improvement Act of 1984 through the pilot process, to be completed in 1991. Major components are described, with details relating those components to the reduction of teacher time and paperwork in the record keeping process. Specific detail is given for the Instructional Management component of CMS because of its potential in managing curriculum and instruction. That potential exceeds its record keeping capabilities.

The hardware, software, and training components are discussed with respect to their impact on the classroom teacher and the services provided by that teacher.

Introduction

The Education Improvement Act of 1984

In 1984, the South Carolina State Legislature enacted the Education Improvement Act (EIA) in an effort to change the direction and impact of the public schools in that State. The EIA is a comprehensive, long-range program to improve the educational opportunities available to students.

The EIA addresses everything relating to the educational process in the public schools. It includes components that give direction to the training of teachers at the undergraduate and graduate levels. Other components evaluate the performance of students, teachers, principals, superintendents, and members of the local Board of Education. Still other components assess and analyze instructional programs, attendance of students and staff, funding, and other aspects of the local involvement in the educational process. Throughout the implementation of the EIA is the common thread that all components must be documented, assessed, and analyzed for effectiveness with regard to meeting defined standards established by the State.

College students wanting to enter the teacher education program at State-supported colleges are given a minimum competency test before being admitted to an education degree program. All education degree programs require five years of study at the college level. The National Teachers' Examination is required before a teaching credential is issued by the State. A degree program is required for certification in school administration.

District and school performance standards are mandated, with the State taking over Districts or schools that do not meet those standards for two consecutive years. Principal and teacher performance standards are established, with remediation required for all who do not meet the State's standards. Student performance standards are established with remediation required for all who do not meet those standards in the basic skills areas (originally reading, writing, and mathematics; science is scheduled to be added to the basic skills standards).

In addition to defining minimum performance standards, the State offers incentive awards to Districts, schools, and teachers who prove to be superior with regard to defined sets of excellence standards. For Districts and schools, the awards can be relatively large amounts of money for use in areas not required by the State's Defined Minimum Program (DMP). For teachers, the monies are personal awards that have no restrictions regarding use.

The Paperwork Problems and Curriculum Concerns

With the implementation of the EIA and its sets of standards, school personnel are made responsible for documenting that standards are met and that remediation is successfully provided in those cases when standards are not met. That documentation includes information concerning district, school, principal, teacher, and student performance.

The first and most immediate problems stemming from the EIA are the collection, analysis, and reporting of data supporting performance and remediation. The management of the paperwork created by the EIA--which, ironically, includes a mandate to reduce teacher paperwork--has been a massive task for district and school administrators and for classroom teachers.

As the State began to monitor the implementation of the EIA, Districts have found that they are cited for items if there is no documentation to support those items. Without such documentation, the State could not confirm that positive educational activities were taking place. A statement has been attributed to one member of a State monitoring team: "If it isn't documented, it didn't happen." Documentation of activities is an essential part of every educator's day. However, research--and word of mouth--indicate that the number of minutes used for teaching and the number of minutes used for planning for teaching has decreased as teachers have spent more of that time documenting the things they have time to do. At the local level, the motto has become, "If it is documented, I didn't have time to do it."

Reducing the Paperwork Problems and Improving Instruction

The State has implemented two major programs in an effort to reduce the paperwork problems experienced at the local district level. The first program, *Osiris*, has been designed to reduce the paperwork at the district and school levels. The second, *The Classroom Management System*, is designed to reduce the paperwork at the teacher level and to share the data stored in the administrative files.

Both programs are computer-based and the State has made the financial commitment to placing the hardware and software in the hands of local educators. In addition to the physical components of the program, the State has committed sufficient financial resources to training district and school staff in the use of those physical components.

The *Classroom Management System* (CMS) provides the tools for reducing teacher paperwork. More importantly, it provides specific tools for helping teachers manage instruction within the district's curriculum, with integration of the basic skills curriculum into other content areas. Similarly, non-basic skills curriculum content can be integrated into basic skills instruction through the management system. That integration can be coordinated through CMS with minimal time and paperwork required of teachers.

South Carolina's CMS is the first attempt, by any State, to standardize and coordinate a management system for all teachers within the State. It represents a major step forward in the operation of the State's public schools within an accountability framework that automates the documentation of needs and services to meet those needs. The impact made by CMS can be significant--if implemented to its fullest potential.

This paper discusses CMS in terms of its reduction of teacher paperwork and in terms of its potential for managing the district curriculum and instruction.

Components and Timelines

The following quotation comes from the "South Carolina Department of Education Classroom Management System Requirements Specifications" document dated December 15, 1987:

> "The success of the Classroom Management System is dependent on the understanding of [its] applicability and [on] utilization of the software by teachers."

Although the title of the document appears to be typical of the bureaucratic phrasing one finds at the State level, the excerpt from the document indicates an understanding of the needs of classroom teachers. It must be noted that CMS is not an administrative package; it is not an instructional package; it is a classroom-level management tool designed to help teachers deal with the management of teaching tasks.

Components of CMS

CMS consists of three components: (1) computer hardware and peripherals, (2) computer software, and (3) training in the use of both.

Computer Hardware and Peripherals. Each school in South Carolina has received a computer and school management software (the *Osiris* package from Tamarac, Inc.). That school management system provides the databases from which CMS will work. As CMS is implemented throughout the State, schools are receiving additional computers that are to be placed in locations accessible to teachers at all times during the working day. The initial ratio of teachers per computer ranges from about twenty-to-one in larger schools to eight-to-one in many smaller schools. As CMS grows, the teacher/computer ratio will decrease dramatically.

The teacher computers will be networked to a fileserver from which teachers will have access to a student database that includes all students receiving instruction in any content area. Each network will include the following equipment:

(1) The fileserver will be an IBM/IBM-compatible computer with an 80386 CPU running at 20 megahertz. Most districts will use the NCR PC916, although many are using the IBM PS/2 Model 80. The fileserver will be equiped with a 286 megabyte fixed disk that will contain the *Osiris* software and files as well as the CMS software and files.

(2) Novell Netware Version 2.11 or above will be used to run the system.

(3) Each CMS teacher workstation will be an IBM/IBM-compatible with an 80286 CPU running at 10 megahertz (or faster). Most districts will use the NCR PC810, although many are using the IBM PS/2 Model 50. Each workstation will also have a 30-megabyte fixed disk to be used for local data storage. EGA color video will be standard for each workstation.

(4) Each workstation on the network will be equiped with a dot-matrix printer that can be accessed from the local workstation without going through the network spool.

(5) Each network will be equiped with a laser printer. That peripheral is necessary because the curriculum and testing modules require graphics capabilities for instructional materials.

(6) Each network will be equiped with an optical scanner. Teachers will be able to administer tests using answer sheets that are "bubbled" by students. Those answer sheets can be scanned and graded by the software.

(7) Each network will be equiped with a telephone modem and a dedicated dataline. Teachers will be able to access resources within the district, the State, and major universities from their workstations.

Specific hardware components will be discussed later in this paper. Those discussions will focus on the implementation of specific program components as a function of the existing hardware.

Computer Software. The CMS software will consist of three major modules, in the initial piloting of the program. Those modules are being developed through a system of specification, review, modification, and implementation. The review process includes a series of teacher reviews and recommendations followed by meetings of district-level coordinators to make specific recommendations to the program developers and programmers.

The software will include the following:

(1) A computer-based gradebook will be provided. Teachers will enter grades from the keyboard in the same manner they enter grades into a paper gradebook. Also, grades can be automatically entered from scanned answer sheets, without the teacher's keying those. Provisions for entering grades from ASCII files stored on diskette will also be included. It will also be possible for teachers to bubble grades on scan sheets and scan those to a specific column in the electronic gradebook. Those last two capabilities will permit teachers to grade items at home and prepare those grades for entry into the gradebook upon return to school the next day.

(2) Test generation and scoring will provide a major tool for determining individual student needs. The test generator will provide access to State administered tests, district-made tests, school-made tests, and teacher-made tests. Test items banks will be developed at each of those levels and will correlated to the State's basic skills objectives. When a test is created, the items will be validated with regard to the measurement of actual needs of students.

Tests can be scored through the use of the optical scanner provided on each school's network. The grades assigned to those tests can be automatically placed in the electronic gradebook.

(3) The instructional management system (IMS), built into CMS, will provide some of the most powerful tools ever developed for educators. The IMS will include assessment and diagnotic tools for determining student needs. It will include tools for the selection and application of instructional activities specifically designed to meet the diagnosed needs of those students. It will link the State's basic skills objectives to the State Testing Program, which includes the Basic Skills Assessment Program (BSAP) and the Comprehensive Tests of Basic Skills (CTBS).

The real power of CMS will come from the IMS component. It will link all areas of the State's basic skills program to the local district's curriculum. That curriculum will be correlated to the State's objectives, to skills, sub-skills, and content skills associated with mastery of those objectives. Resources, including basal texts and supplemental materials, will be referenced in the process of assessment, diagnosis, prescription, and instruction.

Detailed information about the gradebook, the test generator, IMS, and the coordination and integration of curricular areas will be discussed later in this paper.

Training in the Use of Hardware and Software. The State has designated a considerable portion of the financial resources for CMS to the training of teachers to use the CMS hardware and software. The commitment to that training is evident from the quotation cited in the introduction to this paper. The CMS program will not be successful if teachers do not understand its applicability or its utility.

Training will be implemented at the district level by district CMS coordinators who will be trained through the State program. Resources, such as manuals and video tapes, are provided by the State for the use in the district training sessions. Districts may train local teachers in any manner deemed appropriate at the local level. Many districts use graduate-level courses through State-supported colleges and universities to provide core instruction for small groups of teachers in each school. Those teachers then become resource persons within their schools, helping other teachers understand the system.

Examples of some of the training strategies will be given later in this paper.

CMS Implementation Timelines

Ten districts will pilot the CMS program during a three-year period through the end of the 1990-1991 school year. Six schools in each of those districts will be involved in the pilot process. Hardware, software, and training are to be provided by the State, with assistance from the pilot schools and districts. The development of each of those components will be based on review and recommendation from teachers, administrators, and computer-oriented staff.

The hardware will be installed in each school by the end of December, 1988. That hardware can then be used for training teachers in keyboarding skills, software concepts, and general computer use. Word processing and desktop publishing have been identified as desirable training tools that can then remain as real tools to supplement and complement the CMS software.

The gradebook software will be delivered in April, 1989, and will be placed in full use for the 1989-1990 school year. All teachers in the pilot schools will use the electronic gradebook for recordkeeping. The links between the gradebook and the reporting of grades to parents will be discussed in the next section of this paper.

The test generator and the test scoring system will be delivered in April, 1990. Shortly after that, the complete IMS system and the test item banks will be delivered to the pilot districts.

During the summer of 1990, all components of CMS will be put into place. The 1990-1991 school year will be the first full pilot for the complete system, including the test item banks developed by the State.

Following the pilot year, other districts will be brought into CMS and the system will be disseminated throughout the State.

The Typical Scenario

The hardware provided for CMS is simply the vehicle through which the software is implemented. The software provides the real working power of CMS. Before looking at the software details, however, it may be important to understand the potential impact of CMS on the instructional process. (This may be likened to checking the answers in the back of the book before working through the assigned problems. It can provide some direction to our understanding of the details between the problem and the answer.)

The typical CMS scenario may be something like this:

The Summer Setup

During the summer months, all students' schedules are completed within *Osiris*, the school management system from which CMS pulls student information. Teachers' schedules are automatically completed and filed as the students' schedules are completed. From test data stored in Osiris, profile sheets are printed for classes, small groups, and individual students. Those profile sheets provide information about deficiencies related to the State's basic skills objectives.

The Early Weeks

During the first two weeks of school, tests are generated and administered from the CMS test generator. Those results are recorded as the answer documents are scanned or as specific data are entered from the keyboard or composite scan sheets. Those results are printed for each teacher, listing each student with specific skill and sub-skill needs relating to the overall basic skills objectives.

With those extended profiles, CMS will print prescriptions that include curriculum resources and cross-references that can be used in planning instruction. The cross-references will facilitate the integration of basic skills concepts in instruction for all content areas. That is, the prescription may include specific reading resources that can be incorporated into the current social studies and science units being taught by the teacher. Additionally, social studies and science resources may be listed for use in providing reading instruction on the level needed by the students in that teacher's class.

The Regular School Year

When instruction is completed for a specific skill or concept, the teacher can ask CMS to provide test items to measure mastery of that skill or concept. The teacher can personally select items from those displayed by CMS or the teacher can ask CMS to select those items randomly. The selected items will have been previously validated with regard to the skill or concept for which mastery is to be tested. The test can be printed to the local printer or to the network laser printer. Where appropriate, the items will contain the graphic information to accompany the text for those items. The appropriate number of copies can be made and distributed to students.

At the appropriate grade levels, the students can bubble their answers on scan sheets. The scannable sheets can be graded as they are scanned. The teacher will score non-bubbled tests manually, as usual.

Grades determined by the software from scanned answer sheets will be entered automatically into the electronic gradebook, as instructed by the teacher. Other grades can be entered at the keyboard, from an ASCII file containing student ID numbers, the test reference data, and the grade, or from a composite grade sheet bubbled by the teacher. The first of those three options must be done at a network workstation. The other two options can be completed outside the school setting (yes, some teachers will still insist on working at home) and imported through a disk file utility or through the scanner.

The screen image of the gradebook must be as much like the paper gradebook as possible. That will flatten the learning curve for teachers who are learning to use the software. The screen should list students' names at the left, with columns for grades to the right. The top of the screen/page should have space for short labels to identify the test or other graded item. There can be a discription line at the bottom of the screen, in which the teacher can give a detailed description of the skills included on the test. As the teacher moves the cursor from one column to the next, the description line will change to show the appropriate information.

In addition to looking like a paper gradebook, the gradebook software should use only English words, phrases, and sentences in displaying information for use by the teacher. There is no reason, in this day of user-friendly software, to provide anything that will not promote the software-friendly user. Teacher should not be required to learn a new language or a new way of thinking in order to use the CMS software.

That testing, scoring, and recording process can be repeated up to 99 times in a grading period. When that grading period is over, the teacher can confirm that the system has used the proper weights for each graded item and can then finalize those grades. For example, one teacher may weight major tests twice as much as lesser tests and may not count homework assignments in the final grade for students. That teacher will have built that weighting system into his or her gradebook. Another teacher may use an entirely different weighting system (on none at all). That other teacher will have built his or her system independently of that used by the teacher down the hall.

When the grading period is over and the teacher is satisfied that all grades are recorded, CMS can be told to compute final averages (based on the individual weighting systems). A verification report can be printed by the teacher. When all grades are verified, CMS can be told to send those grades, via the network, to the school's fileserver where Osiris will store those in its grade file. When the time comes to print report cards, those are done in the school's office by clerical staff. Report cards are then sent to parents in whatever manner is prescribed by local policy and procedure.

In the grading and grade reporting processes, it should be noted that there is no bubbling of grade sheets or exchanging of grade sheets among teachers. Yet, teachers retain complete control over the grades given to students. That control involves individualized weighting systems, verification (and, if needed, modification) of grades prior to recording and reporting, and modification of final grades at a later time (contingent on local policy).

This scenario is repeated for each grading period, with assessment, diagnosis, prescription, and instructional plans provided by CMS. The recording and reporting of grades is managed entirely by CMS, without taking teacher time to do so.

End-of-the-year reports will summarize student achievement during the school year. Those reports will maintain the appearance of the existing paper formats so that no new learning curve is created for the teachers who must interpret those reports.

Curriculum Management

Test generation and scoring and grade recording and reporting are major takers of teacher time and the CMS contribution to saving some of that time is significant. That time saved can be given to students in the classroom and shared with those pseudo-strangers--also known as family members--with whom those teachers live.

However, the saving of time is only one of the benefits of CMS at the classroom teacher's level. Its ability to manage the curriculum and instruction provides much more power and potential for teacher use. That ability, while assisting in the reduction of non-instructional tasks and time, also provides correlation, cross-reference, and integration of the curriculum in ways not possible without the use of computer technology. With proper planning at the district level, classroom instruction can become more efficient and more effective at the classroom level.

IMS Components

The instructional management system component of CMS provides three basic services to teachers in South Carolina's public schools. Those services include (1) the assessment and diagnosis of individual student needs, (2) the selection and application of prescriptions to remedy those needs, and (3) the record keeping tasks relating the South Carolina state testing program to the basic skills program.

All of those services are dependent on a defined curriculum with specified learning objectives that have been structured along a learning continuum. Each objective along that continuum must, then, be classified in a manner that will identify skills and subskills that can be taught and learned in order to master the objective at the appropriate point along that continuum.

Curriculum Definition. The South Carolina basic skills program identifies such objectives and lists the subskills that must be mastered in order to show mastery of the stated objectives. The skills continuum identifies grade level, the objectives for that grade level, and the subskills for each of the objectives at that grade level.

At the classroom level, additional assessment must be made with regard to appropriate sub-subskills and content skills that would further define the subskills to be mastered by students. The sub-subskills and the content skills complete the "curriculum learning objectives hierarchy" that must be defined, at the classroom level, for each student.

It is at this level of diagnosis, prescription, and recordkeeping that the IMS component of CMS will provide power in planning and implementing instructional programs appropriate for individual students. It is at this level that proper planning by the district can clearly define the curriculum in terms of objectives, skills, subskills, sub-subskills, and content skills. The resulting "curriculum learning objectives database," managed by IMS, can grow with the needs of the district, the district's teachers, and the district's students.

From the curriculum learning objectives database, teachers can have immediate access to reports and outlines that detail those objectives from several different perspectives--and will detail those for a specific group of students or for an individual student.

A summary report will list the general objectives and the subskills of each. This report will provide a summary of the district's curriculum, the continuum along which it lies, and the hierarchical structure of the objectives along that continuum. From this report, teachers can understand the overall scope of the instructional program and the impact that their participation has within that scope.

A student detail report will specify the specific learning objectives for each student served by the teacher. Those objectives will be clearly defined and the IMS record keeping process will provide detailed information about mastery of those objectives. Non-mastery will be further detailed in the student report, listing sub-subskills and content skills that are to be addressed to promote mastery.

A content correlation report can provide information about resources available for use in delivery of instruction. That correlation report can include information about specific student needs in one content area that may be addressed in other content areas, as well. That correlation and integration of curriculum components will add to the efficiency and effectiveness of instruction. That process will provide reinforcement of basic skills from more than one instructional setting (a positive factor from the student's perspective) and will provide more instructional time for those skills within the limited instructional time available during school hours (a positive factor from the curricular perspective).

Assessment and Diagnosis of Student Needs. After a clear definition of each curricular component, students' needs, within that curriculum, can be assessed and diagnosed. The assessment of content skill needs, sub-subskill needs, and subskill needs will be accomplished through the use of test item banks developed at the State, district, school, and teacher level. Each test item in the test item banks will be classified based on the hierarchy of learning objectives and corresponding skills. When an item is selected from the item bank for inclusion on a test, the teacher can be certain that that item has been validated with regard to its measuring some skill related to the overall objective.

Two assessment records will be kept for each student. One of those, the Continuous Assessment Record (CAR), will be available to teachers at all times. It will contain continuously updated information regarding achievement for each student in the class. The Permanent Assessment Record (PAR) will be updated at the end of each school year, using the information from the CAR. (Not that anyone would ever use such an analogy, but it might be noted that the CAR is the vehicle with which students are brought up to PAR.)

The information maintained in the CAR will come from updating the mastery/non-mastery status of students following each assessment. That maintenance can be a clerical function (from the keyboard) or an automatic function (from scanning individual answer sheets or bubbling summary sheets).

From the assessment data, diagnostic reports will be readily available to the teacher. Those can include, but not be limited to the following:

(1) The Student Profile Report will be printed yearly and will include summary needs information for each student.

(2) The Student Assessment Report will be printed periodically to provide information about continuing progress and achievement.

(3) The Student History Report will be printed as needed to provide direction in planning for the immediate needs of each student.

(4) The Class Profile Report will summarize assessment data for the teacher's class, as a whole.

(5) The Group Profile Report will list students by objective, indicating those that have common needs.

(6) The Parent Report will indicate mastery/non-mastery status for the objectives defined for the students instructional program.

Selection and Application of Prescriptions. When diagnosis of student needs is completed, IMS can provide direction in the selection and application of instructional services to remediate those needs. That direction will come from the instructional prescriptions database, which will be built based on the district's curriculum, the available resources, and the location of the student's needs along the learning continuum.

Each item in the instructional prescriptions database will be classified according to defined criteria. When a prescription report is printed for a student or a group of students, it will include a complete description of the proposed prescription, whether the prescription is an activity or a resource, the medium of presentation of the prescription, the current availability of the prescription (including its location within the school or district), the correlation to the learning objective and mastery skills, and the source of the prescription components.

Teachers will have access to several reports from the instructional prescriptions database. The Prescription Summary Report will list prescriptions in the database. That list may include all prescriptions or may include only those selected based on a criterion given by the teacher. The Prescription Detail Report will provide all the classification information for one or more prescriptions, providing detailed information about the applicability of the prescription for a specified objective. The Prescription Student Report will provide detailed information about proposed prescriptions for individual students.

Documentation of Appropriate Instructional Services. In South Carolina, EIA requires the documentation of appropriate services to students who have basic skills needs. That documentation process is extensive and is designed to track student data from assessment and diagnosis to mastery of skills and objectives as measured by the State's testing program. That documentation is a time-consuming task, requiring many hours of teacher and clerical time to complete. IMS will provide the services needed to greatly reduce the manual record keeping and to greatly decrease the non-instructional time used for that record keeping.

Much of the required record keeping will be automatically done through the CAR and PAR facilities in the assessment component. Those records will indicate need and growth as those needs are met. Other record keeping requirements of EIA will also be automatically addressed. Those will include the following, at a minimum:

(1) Curriculum will be correlated to the State's BSAP objectives, including the incorporation of the kindergarten objectives into the readiness program.

(2) The management system for diagnosis, remediation, and placement of students will be standardized and evident.

(3) Evidence of parent notification will be maintained for the readiness program, the testing program, the identification of instructional needs, and the placement of the student in appropriate programs.

(4) The status of students with regard to the Exit Exam will be automatically maintained.

(5) For each student, IMS will document the curriculum objectives that remain unmastered.

(6) Program descriptions, components, and timelines will be documented for all remedial and compensatory programs.

In addition to the above record keeping tasks (all of which will be automated to some degree), the IMS module will maintain records regarding the results of the State's testing program and will provide a variety of reports that can be used in interpreting those results and in applying those results in constructive ways to the instructional process.

In the final analysis, the records kept in IMS will document student performance and achievement. Those records will be updated at the end of each school year, leaving a clear record along the learning continuum for the beginning of the next year's instruction. That record keeping process is currently done manually by the teachers in South Carolina. (In Greenwood School District 50, teachers keep those records on "Make-A-Difference" cards. The acronym for those cards is also an appropriate adjective for those who must manually manipulate them.)

Cautions and Concerns

Before closing, it is appropriate to point our some concerns about CMS—specifically, about its IMS component. Because the IMS component of CMS offers so much power in managing curriculum and instruction, it may be misused in several ways. Those misuses are not an inherent function of IMS itself; they are more a function of the educational process (whether automated or manual). The following cautions relate to possible perceptions that may develop about the purpose of IMS.

The first caution concerns the development of curriculum components that are designed to fit the perceived uses of the management system. A good management system should fit the previously designed curriculum--the curriculum should not be designed to fit the management system. The latter is the same as "teaching the test"--at a higher level, of course.

Proper use of CMS is predicated on the existence of the appropriate curriculum for the individual school district. When that curriculum exists and it is properly designed to offer a skills continuum through which students pass, its components can be coded within the management system so valid data are collected and reported from within that system.

The second caution concerns the potential limiting of the CMS to the basic skills areas (in South Carolina, those are reading, writing, mathematics, and science). Its management capabilities go well beyond those basic skills areas. The CMS Requirements Specifications document states: "The Instructional Management component of the CMS is in no way intended to be restricted to usage in a remedial environment."

However, most state testing programs (including South Carolina's) measure achievement only in the basic skills areas areas and report that achievement in terms of meeting or not meeting a set of minimum standards. There may be an unnecessary focus on those content areas with regard to the development of test item banks, diagnostic tools, prescriptions, and referencing instructional resources.

IMS has the capacity to assist teachers as they provide an integrated instructional program, using resources from several content areas to teach basic skills. It can assist in integrating basic skills instruction in the provision of services in other content areas. Its capacity to do so is limited only by the planning at the district level and the provision of the cross-references linking all those content areas.

Not only will IMS address the basic skills areas, it can be used to manage the curriculum for all content areas. Additionally, it can be used to correlate and integrate those content areas so teachers can get "double duty" from each instructional activity they provide in the classroom.

Conclusion

South Carolina's CMS pilot program is an attempt to provide management tools to teachers so that record keeping, planning, and delivery of instruction can be automated as much as possible. CMS is not an administrative or instructional program.

CMS includes three major software components: (1) the instructional management system, (2) the test generation and scoring system, and (3) the electronic gradebook system. Each of those is coordinated with *Osiris*, the State's school management system. That coordination results in class rosters for gradebooks, automatic printing of report cards, electronic record keeping for grades and transcripts, and other features that reduce teacher paperwork and redundant record keeping.

The hardware, software, and training are provided by the State, as part of the Education Improvement Act of 1984.

Many of the components of CMS are generic in nature and can be exported to other states or to individual districts. Those components are available commerically from the vendor with whom South Carolina has been working. Many components are specifically designed to address the State's record keeping needs and are, thus, not readily exportable.

South Carolina has recognized the educational addage that "every complex problem has a simple solution--and it is wrong." The CMS focus is a sophisticated approach to solving complex educational problems--problems for which there are no simple or short-term solutions.

Bibliography

Hill, Jack D., Jr., Ph.D., "The Pathways Project: South Carolina's Bold Attempt to Reduce Teacher Paperwork Through Microcomputer Technology"; *T.H.E. Journal*; November, 1987; pp. 87-91.

Tamarac Systems, Inc.; "South Carolina Department of Education Classroom Management System Requirements Specifications"; December 15, 1987

Tamarac Systems, Inc.; "South Carolina Department of Education Classroom Management System Requirements Specifications"; July 25, 1987.

Tamarac Systems, Inc.; "South Carolina Department of Education Classroom Management System Gradebook System Specifications"; August 26, 1988.

BEYOND DRILL AND PRACTICE: USING MICROCOMPUTERS WITH LEARNING DISABLED STUDENTS

Susan Jo Russell
Rebecca B. Corwin
Janice Mokros
Peggy M. Kapisovsky
TERC
1696 Massachusetts Avenue
Cambridge, MA 02138

ABSTRACT

In this session we will report on a variety of promising practices largely based on the use of learner-centered software with special needs students. Learner-centered software will be defined, examples of learner-centered software will be presented, some promising uses of the microcomputer with special needs students will be highlighted, and resources will be identified.

The advent of the microcomputer in special education has raised many questions about "best" uses of the microcomputer with children. Nowadays most school settings include some access to the microcomputer, but as yet its use is still in the formative stage. The most common use of the microcomputer with special needs students is drill and practice; that is not the focus of this presentation. Rather, we will explore other uses of the microcomputer so that the range of computer use can be expanded. A broad repertoire of software uses with special needs students can be adopted by regular classroom teachers with special needs students mainstreamed into their programs as well as by special needs teachers.

The project upon which this report is based was funded by the Office of Special Education and Rehabilitative Services, and included both in-depth work with resource room teachers in Massachusetts and a national study of promising practices using the microcomputer with special needs students in a variety of settings. The knowledge base of practitioners and researchers is blended in this project, which will culminate in the publication of a book entitled *Beyond Drill and Practice: Using Learner-Centered Software with Special Needs Students*, slated for publication by the Council for Exceptional Children sometime in August 1989.

The presentation will be helpful to regular classroom teachers as well as those who deal full-time with students with learning difficulties.

NETWORKING AND TELECOMMUNICATIONS

Frank DiGiammarino, Beth Lowd, Bruce Mallory, Amy Wagner
Donald F. Orcutt, Donald M. Weinberger
Dennis Sayers

ABSTRACT: LEXNET—Telecommunications from Every Classroom

Frank DiGiammarino, Director for Planning; Beth Lowd, Computer Resource Person; Bruce Mallory, Information Science Teacher; Amy Wagner, Science Program Manager, K-8: Lexington Public Schools,1557 Massachusetts Avenue, Lexington, MA 02173.

We will show how to use your cable television drops in each building or classroom as a connection to a central modem and telephone line. If, as in our case, you have a drop in each room, it means that any class can have access to a telephone line right in their classroom. All you need is a computer on a cart with a special modem that you can wheel into the room when it's needed.

Our state-funded model has involved setting up the system, creating a bulletin board that allows students within Lexington to share science data easily, training teachers and library media specialists to use the system, and piloting a science unit on water quality in grades 5, 6, 8, and 9. The system has also been used to introduce teachers to the use of a bulletin board for informal communications and to share some data about the recycling of paper in our community.

We will hand out a project summary (describing the training and activities) and a sheet of technical specifications for the stystem, and we will show a brief videotape of the system in action.

ABSTRACT: Telecommunicating Between Schools Utilizing an Intermediate Unit Electronic Bulletin Board in the Development of Newspapering Skills among Gifted and Regular Sixth Graders

Donald F. Orcutt, Donald M. Weinberger, Indian Crest Middle School, Souderton, PA 18964.

An elementary gifted program can serve as a hub through which certain educational and social needs of regular and gifted students can be addressed. Near Philadelphia, Pennsylvania, teachers of gifted and regular sixth-grade students in the Souderton Area School District and personnel from the Montgomery County Intermediate Unit #23 have joined forces to implement a program which applies skills in the areas of language arts and social awareness, through the use of telecommunications.

We have developed a multi-faceted program designed to encourage all children to make the most of their individual talents and abilities in living life to the fullest. They have been guided in the development of an 'up with kids'-style bimonthly news magazine that advocates kids making the most of themselves.

The students in both groups have related various positive happenings concerning children in our local communities. Articles designed to help others find interests and expand their capabilities were presented by fellow classmates in the regular and gifted programs. A community calendar of doings for kids in the area was also included in each issue.

Through the use of the MCIU #23 Electronic Bulletin Board mail system, classrooms were able to network with one another to upload and download articles and messages relating to the publication.

The gifted students served to coordinate the production and publication of the magazine, "Excell".

In the regular classroom setting, the program was integrated into the S.A.S.D.'s language arts curriculum in a manner that sought to develop the following skills.

I. Process Writing Skills: prewriting, rough draft, revising, editing, and publishing
II. Journalism Skills: impact, timeliness, relevance, consistency, honesty, balance of fact and opinion
III. Specific Reporting Skills: interviewing, reviewing, editorializing, and composing features
IV. Computer Skills: selecting software, graphics/text integration, word processing, transferring files, and networking

ABSTRACT: Telecommunications and Second Language Learning: International Cross-Cultural Connections

Dennis Sayers, New England MRC, University of Hartford; George Migliarini, Page Schools, Costa Mesa, CA.

This session presents the work of Project ORILLAS, an international Sister Class network to encourage language learning in bilingual, ESL, and foreign language classes, offered through the McGraw-Hill Information Exchange (MIX) computer network. Since 1985, Orillas has put second-language students in Argnetina, French-speaking Canada, Mexico, Portugal, Puerto Rico, and the United States in direct contact with native speakers to develop not only basic language skills, but cultural awareness and communicative competence. Based on the pedagogy of French educator Celestin Freinet, ORILLAS students exchange more than penpal letters, working on collective projects including the sharing of digitized photos, synthesized voice, and desktop publishing files, as well as driving robots on other continents via modem. Demonstrations of these activities will be offered.

GETTING THE MOST OUT OF LARGE PROGRAMMING PROJECTS

James L. Wolfe
Computer Science Department
Indiana University of Pennsylvania
Indiana, PA 15705

Abstract

This paper discusses the tasks a professor must perform and the issues s/he must consider in using a large programming project in a course. Preparation tasks, methods of administration, evaluation considerations, and project structure are examined. Recommendations are given for making a project successful for both the students and the professor.

Introduction

Having a large programming project in a course puts a strain on both the students and the professor. Students must adjust their approach from the typical projects, which require them to work individually and to write a two- or three-hundred line program in perhaps two weeks, to an extensive project, which may require student partnerships and a two- or three-thousand line program lasting all term. Professors are faced with enormous problems of sequencing lecture material, controlling student activity, and evaluating complicated student submissions.

Nevertheless, a large programming project can be one of the best ways to reinforce the concepts presented in lectures. Students get a chance to apply complicated ideas and algorithms rather than just hearing about them. Large projects are particularly useful in systems programming in which so much is going on concurrently that lectures must focus on one area or facet at a time. A large project can integrate the concepts and make the big picture clear. [2,3,5,6]

Here, we concentrate on systems programming projects (writing operating systems, compilers, assemblers, disassemblers, linkers, macro processors, editors, etc.) as examples of large projects that may be done in one term. We also assume that terms are 12 to 15 weeks long and that only one large project is assigned during a term. (Projects lasting two terms have similar needs and requirements. [4]) Discussion and recommendations made here are based on experiences with eight large systems projects that have been developed and used multiple times at Indiana University of Pennsylvania.

Setting Up the Project

There are several important issues that a professor must consider while creating a large project. These issues either apply only to large projects or must be resolved more fully than for small projects.

Real vs. Simulated Computer

Can the computer(s) available at the school be used directly or must a computer simulator be devised or acquired? For an operating system project, this is a question of whether or not a machine dedicated to the course can be found. Generally, operating system development cannot occur on a machine at the same time as other software is being used. (There are exceptions. [2]) On a language translation project, the object code form for a real system is at issue. It may be very difficult for a student compiler or assembler to produce an object code file that can be linked and executed by the system's software.

A simulated computer gives the professor much greater control over the project environment. The simulator can be made as elaborate or as simple as desired and can include a variety of features. However, there is a cost - the professor must write (or acquire) the simulator software. Pedagogically, simulated computers are to be preferred over real ones because of the control they provide. The remainder of this paper assumes that simulated computers are used.

Student Groups

Should a student work alone on a project or should s/he have one or more partners? A reasonable rule of thumb is to use the size of the professor's solution to the project to determine the size of the student working group. Experience at IUP has shown that students can individually handle projects requiring as much as 700 to 900 lines of Pascal code; students in pairs can handle projects of as much as 1500 to 2000 lines of Pascal code; and students in groups of three can handle projects of as much as 2500 to 3500 lines of Pascal code. Experience has shown also that even larger projects cannot be completed in one term and that groups of more than three students are not conducive to having everyone contribute.

For groups of two or three, the project must be arranged so that cooperation between members of the group is facilitated, encouraged, even required. Portions or features of the project should lead the students to discussion so that they can come to an agreement on the techniques to use, the algorithms to employ, and the interfaces between individual student contributions.

Scope and Flexibility

How large is "large?" Ideally, the professor would like the large project to encompass everything being taught in the course. Unfortunately, the requirement of insuring that the project can be completed in one term forces some restraint. The issues become what to leave out without making the project unrealistic and what to put in without making the project undoable.

For an operating system project, the professor wants to include memory management, file management, CPU scheduling, device control, concurrency, command language interpretation, etc. However, should file access control be implemented? Should batch jobs be supported? Should total residency, swapping, or demand paging be used? The professor must determine the relative benefits of including features and weigh them against the cost of having the students implement the features.

There must be a reasonable amount of flexibility built into a large project. Students must feel free to choose among several techniques for implementing a feature. Students should also be given some leeway in choosing optional features to implement. These options must be handled carefully; a student (even a hot- shot programmer) must be dissuaded from trying to implement some sophisticated feature that could double the size of the project.

Support Tools

What programming tools should the professor provide the students to help them with debugging? A simulated machine does not usually provide an interactive debugger such as those on a real system. Yet, the professor must provide the students a way of determining what the simulated machine is doing with their software. For an operating system project, the simulator should be able to trace such things as I/O transfers, program executions, memory usage, and interrupt actions. For a language translation project, the simulator should be able to show register and memory contents during the execution of a program.

Professor Preparations

Long before the students are handed a large programming assignment, the professor must develop the materials necessary to administer the project. If the professor is lucky, a simulator that meets the needs of the course may be available. Otherwise, the professor must create the simulator from scratch. Either way, there must be sufficient documentation available for the students to understand the machine being simulated.

For an operating system project, the simulator documentation must include a description of the simulated hardware, of how it appears to function, and of the error messages and reports produced by the simulator. For a compiler project, the documentation must include a description of the instruction set for the simulated machine, of the object file form, and of the way in which linking is handled. A description of debugging tools is always needed. Generally, the professor must generate at least some of the documentation, even for an acquired simulator.

Devising the project is part of the preparation phase. However, the professor must do more that just devise; s/he actually must DO the student project. The professor begins by writing the project description and defining the requirements to be placed on students. Then, the professor should try to use the description and requirements as s/he expects the students would to solve the problem. If the professor is fortunate, s/he may find an independent-study student to try out the project in this way, relieving the professor of actually writing the project program.

By going through this trial run, the professor can detect weaknesses in the project description, the simulator documentation, and even the simulator. S/he can determine where the project bottlenecks are, i.e., what parts take extra care in programming, what parts take extra explanation, and what optional features should be recommended against. The professor may find that auxiliary support programs are needed to help the students with evaluating their project programs, that tighter controls are needed on the form for inputs or outputs, or that the simulator must do more error checking or tracing. The professor may even discover that the project is too difficult or too easy and must be adjusted.

Any software that the professor writes to assist the students during the project must be robust and clearly documented as to its use. For example on a compiler project, the professor may provide the students with a program that generates the parse tables from a student-written grammar. Such a program must be usable by the students without forcing all students to write exactly the same grammar. It must handle whatever grammar the student writes and either generate a parse table or state what is wrong with the grammar.

One could argue that it is not really necessary to have a simulated computer on which to run the compiled or assembled programs, that the students and the professor could determine the success of the translators by examining the object code. Thus, the professor would not have to generate the simulator. However, such an approach would deprive the students of the sense of satisfaction that they gain by seeing the translated programs execute. Having no simulator also deprives the students of a debugging tool in that the simulator is the final determiner of whether or not their translator did its job. Not being able to execute the programs makes the professor's evaluation of a student project more difficult as well.

Administering the Project

A large project represents a delicate, personnel management task. Students typically remain a little wary of their partners; they worry about having enough information and time to complete the project. The professor must be prepared to reassure, to arbitrate, to cajole, and to provide leadership, as well as to instruct.

Classroom Issues

It is essential that the professor carefully plan a course's lectures so that students can digest the concepts and address the implementation issues of the project in a timely fashion. For example in a compiler course, the students cannot get started until they understand lexical analysis. However, while they are working on the lexical analysis phase, the professor can present the syntax and semantic analyses concepts so that the students are ready to begin implementation of syntax-directed translation without delay. Similarly, code generation may be presented while syntax-directed translation is being implemented. This sequence of topics is natural to both the teaching of compilation and the development of the project.

Design must be stressed. If the professor provides the overall design, the students must be required to provide the design details BEFORE they leap into coding. If the students are responsible for the whole design, software engineering techniques should be used in the writing and reviewing of the design specifications. Informal student©written designs should be regarded as a last resort, not as convenient short cuts.

The professor should expect to devote some class time to providing general help with the projects. Some students need more substantial hints than others. Most students will need to ask questions, many of whose answers will benefit the whole class. In Q/A sessions, the professor can catch students before they wander off into programming quicksand. The answers to questions such as "What do you mean by ...?", "Can we do ... this way?", "Will there be data that ...?", and "Is it OK to ...?" can help direct the students along paths that lead toward project completion and away from dead ends holding many frustrations.

Forming Working Groups

There are a variety of ways to select the students that form a working group for a project. Each has associated problems.

1. Let the students choose their own partner(s). Often students choose partners of roughly equal academic ability because they are socially stratified on achievement levels. Consequently, high achievers have an even easier time and low achievers have an even more difficult time. Also, unpopular students and those that are not well known have trouble finding partners.

2. Assign the students to partnerships randomly. Partners that are badly mismatched (from an academic point of view) tend to have a stormy partnership. The weaker student cannot pull his/her weight and is frustrated; the stronger student may try to do more than his/her share and resent doing this. Personality and work habit mismatches can also arise in random assignment, e.g., the self-starter with the procrastinator.

3. Allow students to work alone or with a partner (chosen by 1. or 2.). Allowing some students to work alone creates a disparity in the workload for the students. Students, who would benefit by having one, do not choose a partner because they "don't know anyone"; or they fear who will be assigned to them. Students with the ability to do the project alone choose a partner and both breeze through the project.

4. Assign students to partnerships based on the professor's knowledge of them, trying to match those that would seem to be compatible. Professors often have little basis for selecting partners this way. Also, students may blame the professor for mismatches and may become even more resentful and rebellious than with random selection.

Despite the problems, each method of selecting students for groups has its uses, depending on how the project is organized. However, random selection is perhaps the fairest method; and its problems can be overcome using the proper control technique.

Maintaining Control

If the large project can be broken into several segments, the professor can repeatedly assign random partners for each segment and overcome some of the associated problems [3]. Along with reforming the working groups for each project segment, the professor needs to be prepared to assign individual scores to each member of a partnership. Handling the scoring in this way can remove some of the animosity felt by partners of different academic abilities. One technique to facilitate individual scoring is to require each student group to provide a statement of what each member will do as they start work on a segment. This contract along with a statement of who did what at the end of the segment can be used to identify who is and who is not pulling his/her weight. Then, the individual scores can be determined.

Regardless of how well the partnerships work out, the professor must plan on some (perhaps a majority) of the groups not completing the project. Segmenting the project combined with the professor's own implementation of the project can help greatly with such failures. If a group fails to complete segment X and subsequent segments depend on segment X, the professor may be able to let the group use the professor's own segment X and proceed from there. Thus, non-completion of any one segment does not necessarily lead to catastrophic failure. The use of randomly reassigned partners for the next segment combined with use of the professor's code can give the students a sense of having a fresh start on the remainder of the project.

Gauging Success

There is much more to determining the success of a project than calculating the fraction of the class that produced a working program or looking at the average score for the project. The professor must be conscious of how hard s/he had to push the students to get the results. The professor must estimate how much pushing was needed to overcome the intimidation of the task size and student's procrastination. Too little pushing leads to no one completing the project. Too much pushing can lead to resentment or rebellion by the students; they must be convinced that they can do the project, that they have the necessary information, and that they will benefit from it.

It is highly beneficial to get student feedback (preferably in an anonymous form) on their perceptions of the project. Certainly there will be some petty complaints; some students may reject the whole idea of partnerships on projects; however, there may be a pattern in the complaints. Such a pattern could be a symptom of a problem that the professor must address. Student feedback may even provide some helpful suggestions for the next time around.

Only when a project is used a second or third time can the professor make the assignment with clear expectations of what the students can do. Only with experience can gauging how well a project is going and evaluating the final results have a firm basis; then there is a project history with which to compare.

Reusing a Project

Too much work goes into the creation of a large project for the professor to be satisfied with using it only once. Yet, simply reusing a project the next time a course is taught is asking for plagiarism trouble. Experience has shown that there is much less danger of plagiarism in large programming projects than in small projects. This is probably due to the fact that large projects are typically found in senior-level courses and that such courses are populated by students who have developed a good set of programming ethics.

Nevertheless, it makes sense to put some changes in a project each time it is used. The changes may be small but can be significant in terms of how they are perceived by the students. For example, a compiler project may involve a language with multi-dimensional arrays one time and records another time, a "while" structure one time and a "for" structure another time, or recursive procedures one time and FORTRAN-like subroutines another time. The key is that the original project design must be flexible enough so that the professor can include or exclude a variety of requirements. The varying requirements can adjust the overall scope of a project, as well as generating sufficient perceived differences to exclude plagiarism.

Sometimes it is possible for the professor to use one project to develop other projects. For example, the professor may have supplied a linker and simulated machine in order to execute the programs acted on in a student compiler project. In other courses, the professor could use object programs produced by the professor's version of the compiler as data and assign the students the project of programming a linker, a disassembler, or the simulated machine itself. Such projects are much cheaper to develop than the original compiler project. Much of the documentation can be reused or slightly modified; little or no additional programming may be required of the professor.

Conclusion

The basic conclusion is a cliche, "That which one gets out of a project is in proportion to what one puts into it." The professor must emphasize preparations and administrative control. The students must be dilligent and cooperative. Although not easy to devise, the best large projects are segmented, so that student progress can be monitored, and are well-documented and supported, so that student confusion and frustration are minimized. Providing these features takes a lot of work on the professor's part.

Of the eight large systems projects that have been used in 18 course offerings on four simulated and two real machines at IUP, only the latest two projects have the majority of the ideal attributes described in this paper. The development of these projects has been a refinement process in terms of improving project organization and student working group composition. The latest project uses project segments, random reassignment of groups for each segment, use of the professor's code on non- completion, group member committment contracts, and individual scoring of group members. Student complaints about the groups are greatly reduced; some students even praise the projects for being like job situations. Everyone is getting more out of these projects.

References

1. Donaldson, J.L., "Teaching Operating Systems in a Virtual Machine Environment," *SIGCSE Bulletin*, 19, 1 (Feb 87), 206-211.

2. Noonan, R.E., "Compiler Construction Using Modern Tools," *SIGSCE Bulletin*, 18, 1 (Feb 86), 109-113.

3. Petersen, C.G., "The Evolution of a Project-Oriented Compiler Writing Course," *SIGCSE Bulletin*, 17, 1 (Feb 85), 116-119.

4. Sathi, H.L., "A Project-Based Course in Compiler Construction," *SIGCSE Bulletin*, 18, 1 (Feb 86), 114-119.

5. Shay, W.A., "A Project for Operating Systems Simulation," *SIGCSE Bulletin*, 18, 1 (Feb 86), 289-295.

6. Wolfe, J.L., "Operating System Projects on Two Simulated Machines," *SIGCSE Bulletin*, 19, 1 (Feb 87) 212-216.

TEACHING MACINTOSH ASSEMBLY LANGUAGE TO COLLEGE FRESHMEN

W. D. Maurer
Department of Electrical Engineering and Computer Science
The George Washington University
Washington, DC 20052
202-994-6569

Abstract

This paper describes the challenge of teaching the assembly language and machine language of the Motorola 68000 to students who have had Pascal, but not necessarily data structures, logic design, or computer graphics. Due to the unusually complex instruction set of the 68000, it is easy to fall into the trap of using a concept before it is defined, or defining a concept before the student has enough background to understand how that concept would be used. Thus a logical ordering of the syllabus has been of unusual importance in meeting the challenge of teaching this material at this level.

Introduction

In June 1988, the Department of Electrical Engineering and Computer Science of The George Washington University terminated its teaching of the assembly language of the Apple II and substituted that of the Apple Macintosh, using the MPW (Macintosh Programmer's Workshop) operating system. This system includes an assembler with a rich collection of directives, as well as Pascal and C (all of which can mutually call each other) and a machine-level debugger. It is available to all our students through the Waterloo MacJANET network of 30 Macintosh Plus and two Macintosh II computers. Typical course loading in the fall semester is 85 students.

An initial textbook search for this course appeared to indicate that all of the available textbooks for teaching this material were inappropriate in one way or another. Those which were actually texts, rather than reference material, seemed to require that the students already know either data structures, elementary logic design, or computer graphics. Yet the curriculum of this department specifies assembly language in semester 2 (of 8); data structures in semester 3; logic design in semesters 4 and 6; and computer graphics in semester 7. While this created no difficulty when the assembly language was that of the Apple II, it posed a considerable challenge when the Macintosh was used. Accordingly, this author, who had earlier produced an Apple II assembly language text [1], undertook to provide a set of notes for Macintosh assembly language.

Our educational design goals were as follows:

(a) We were first and foremost concerned with assembly language as a *real language*, in which to write real programs, and not as an exercise of mainly theoretical interest. Over the years, the appropriateness of this has been a matter of considerable debate. Assembly language programs are far harder to debug than programs written in any other language. Many commercial mainframe programs, written in assembly language, could have been written in Cobol with a far lower total cost of operation, when both production and maintenance are taken into account. Also, in recent years, the development of the C language has shown that even an operating system, such as Unix, can be written in an algebraic language if that language is designed for fast execution. These three factors, among others, contributed to a climate of opinion opposed to the teaching of assembly language, except as a survey course in which the students write one or two programs, or as an adjunct to courses in logic design.

Yet there are good reasons for knowing how to program in assembly language. Certain programs must be written in assembly language; some because of speed considerations, others because of execution on very small systems that cannot support the run-time environment needed by compiler output. (Cross-compilers are of no help here, since the output of a cross-compiler still needs a run-time environment.) More generally, a program will often run much faster if only its most heavily used sections are recoded in assembly language. This, in fact, provided us with our first specific design goal: to teach students to write ***assembly language subroutines of MPW Pascal programs***, since all the students were familiar with Pascal at this point.

(b) Regardless of what machine is involved, the students should learn the *whole* machine language, and the *whole* assembly language. This is *not* to say that they should memorize even half of the assembly language syntax, and they should not memorize *any* of the machine language. It does mean, though, that they should be taught enough to be able to look up, in tables, whatever they need. The tables, ideally, should cover every operation code, every addressing mode, every data size, every mnemonic, and every directive.

On the Apple II, this was easy (see, for example, [1]); on the Macintosh, it appeared so daunting that considerable doubts were expressed as to its mathematical possibility. The instruction repertoire of the 68000 includes 104 instructions (46 of them conditional, of the form B*cc*, DB*cc*, and S*cc*) involving up to 21 addressing modes each, and, in one case (MOVE), involving $21 \cdot 15 = 315$ addressing mode combinations. Many of these instructions, in addition, are available in three data sizes, multiplying the total number of combinations by 3. The MPW Assembler involves 48 directives, not counting the macro assembly and conditional assembly features. In addition, there is the 68881 floating point chip, which involves seven data sizes (the three integer sizes plus four more for real numbers) and another 138 instructions (94 of them conditional).

In view of this, the degree to which we have attained our goal is considerable. Macros and conditional instructions, as it happens, had never been taught in earlier incarnations of our first assembly language course; we now introduce just the bare minimum needed to understand the idea of a macro. Also completely omitted are the system calls, most of which, on the Macintosh, have to do with graphics, which the students will learn later. Our ability to teach almost all of the remaining material is based on the fact that there is far more rhyme and reason in the 68000 than is apparent on first encountering the machine.

(c) For everything that we teach, we also, ideally, ought to explain *why* it is there, and what it is used for. This is particularly challenging in connection with the addressing modes. Manufacturer's literature on the 68000 lays out each of the 21 addressing modes for an `ADD` instruction, for example, but is rather sketchy concerning the use of each one. It is clear that most of them have to do with subscripted variables; but why that many? Also, one searches in vain for the "standard indexed addressing mode" available on most computers (absolute long address in the instruction, together with a reference to a register containing the index); it is simply not there. What should be substituted for it, and under what conditions?

(d) Each of the tables to which the student refers should, if possible, be condensed to a single page. One of the most frustrating experiences in the professional life of an assembly language programmer is to see a single concept illustrated by multi-page tables, which make it very difficult to compare one instruction with another. It is easy to prepare single-page tables for the effective addresses, the conditional instructions, and the register and flag capabilities. It was considerably more difficult to do so for translating assembly language to machine language, for identifying which addressing modes are usable with which instructions, or for instruction cycle times, but this has at length been done.

Our instructional semester is made up of 14 weeks, of which one is given over to midterm examinations. The assembly language course, Computer Science 147, covers one subject in each of 13 weeks; for each week, we have a handout averaging 11 pages in length (single spaced, using the 12-point Times font on the Macintosh). We now describe the material covered in each week.

(1) **Introductory.** Many assembly language courses start with a week's review of Pascal (or whatever other algebraic language is being used — BASIC, in the case of our Apple II course). We have used this week for five tasks. The student should get a review of Pascal; a short introduction to what assembly language and machine language are, and how they relate to algebraic languages; an introduction to the Macintosh (for those who have never used it); an introduction to the MPW system, and in particular to MPW Pascal; and, finally, an introduction to the **go to** statement and to the ways in which **if-then-else**, **for**, **while-do**, **repeat-until**, and **case** statements can be expressed in terms of equivalent collections of assignment, **go to**, and conditional **go to** statements. This last is necessary because many entering students today know *only* Pascal (not BASIC or FORTRAN) and have never used the go to statement in their own programs (in fact, have been required by their secondary school teachers *not* to use it). Thus the essential character of machine language as a language without structured statements is new to them and may take some getting used to.

During this first week, the students write a Pascal program which is given in the class notes, character for character. It is important that they type this in themselves, and that they not simply copy it from the instructor's file, as this gives each student a feel for the mistakes which he or she makes most commonly. This Pascal program is compiled and run, and the output printed. Since MPW Pascal has no commands which produce output directly to a laser printer, we write the Pascal program in such a way that it produces output on a file with suffix dot-LST (such as **tsub.lst**), in addition to screen output. The file can then be printed by opening its window and using a Print Window command. Also, the Pascal program is written with a very short Pascal subroutine (which does just the single statement U := V–W). This subroutine will shortly be rewritten in assembly language.

(2) **A first week's worth of assembly language.** The choice of material to be introduced here is dictated entirely by the goal of *having the students write an assembly language subroutine of a Pascal program*. The subtraction program above (U := V–W) is, in fact, written *for* the students and presented in the class notes, with full explanations of every instruction and every directive that is involved. It is then explained to the students how to modify their Pascal program so as to call a procedure written in assembly language. (By the way, although this is not standard Pascal, it seems to be done in the same way in most extended Pascal systems, namely by using the keyword `EXTERNAL;` after the procedure heading.) The first homework problem is now given: in addition to both the Pascal and assembly language versions of **sub**, the subtraction program, they are to write Pascal and assembly language versions of a program to multiply a number by 23, using addition only (and whatever optimizations they can discover). This is due at the end of the fourth week.

Many of the directives of MPW are essential to this task. Pascal parameters are kept on the stack. When the assembly language program gets control, the stack will contain the return address and the parameters in *reverse* order; it then proceeds to use the `LINK` instruction (for Pascal-style automatic storage allocation), which saves register A6 on the stack. In order to make reference to the parameters, rather than teaching the students to calculate displacements for each parameter, we show them how to construct a parameter record template, making use of MPW Assembly Language facilities originally developed for defining Pascal-style types in assembly language programs. Such a template does not actually allocate any storage; this is supposed to be done by DS (Define Storage) statements making reference to the given type, but here it is exactly what we need because the parameters, together with the return address and the value of A6 saved by the `LINK` instruction, look exactly like a record except for the fact that they take up no actual space in the given program (because they are on the stack, and then only temporarily). The students are also shown how to move data into and out of parameters, both **var** and non-**var**, and are given a specific sequence of instructions for ending a program (to be explained in Chapter 6.)

(3) A first week's worth of machine language. On the 68000, unlike the 6502, we do *not* consider it necessary that the students know even the rudiments of binary arithmetic before they start understanding their first assembly language program. This is because many extremely simple instructions, such as add without carry and subtract without borrow, are missing on the 6502, whereas on the 68000 every necessary elementary operation has its corresponding instruction. Indeed, if it were not for the massively complex instruction set, addressing modes, and machine language, 68000 assembly language would be an unadulterated joy to teach: it has all the arithmetic operations (+ – * / and unary –), all the logical operations (**and**, **or**, exclusive **or**, and **not**), all the relations (= ≠ < > <= >=) in both signed and unsigned versions, all the varieties of shifting (arithmetic, logical, and circular; left and right; register and memory; constant length and variable length), and so on.

In this week, then, the students learn the usual material about binary and hexadecimal numbers, twos' complement, bytes, words and long words, registers, and the fact that signed addition and subtraction are the same as their unsigned equivalents (and proofs of these facts). They also learn the ***elements*** (though not the details) of instruction codes, instruction code fields, absolute and relative addresses, character codes, word boundaries, immediate addressing, and program counter addressing. The pitfall here would be introducing the addressing modes too quickly; these need to be illustrated with usable examples, and they are postponed until week 5.

(4) Subscripted variables. We now start the long process of introducing the addressing modes. In Chapter 2, the students learned the modes **Dn**, **An**, **#k**, v(A6) (for making reference to parameters) and **(An)** (for storing in **var**-parameters). They now learn, one by one (and are given examples of the use of): post-increment addressing, **(An)+**; pre-decrement addressing, **–(An)**; index-displacement addressing (but *without* the displacement, for the moment), **(An,Rm)** (with **An** pointing to the start of an array and **Rm** containing the index, possibly multiplied by 2 or 4); displacement addressing for constant subscripts, **k(An)**; general index-displacement addressing, **k(An,Rm)**, for subscripts of the form **v±k**; and program-counter addressing, both indexed and unindexed. At the same time, they learn about arrays T starting from T[1] rather than T[0] and the additional displacements associated with these (the use of T–1(**Dn**), for example, for a character array of this type, where **Dn** contains the index). This, in fact, covers all the addressing modes except for status register addressing (postponed to week 7), the dot-L modes such as **k(Am,Rn.L)** (taken up in week 10), and absolute short and long (postponed to week 12). We feel that it is far more important to learn uses of each addressing mode *as it is introduced* than to have addressing modes "thrown at one" as a subject of mystification. Two-dimensional array processing is postponed to week 11.

In this chapter we also take up our second programming assignment — a sorting program. The students are initially given a straightforward interchange sort, in both Pascal and assembly language, which they are to copy and run, much as in the first assignment. They are then given an insertion sort in Pascal, which they are to translate into assembly language and run. So far the student has done no actual assembly language program design (this will start in week 6), but the two weeks of hand-holding have been necessary as the students familiarize themselves with the MPW system and with assembly language concepts.

(5) A second week of machine language. Only now do we get to the main business of machine language: effective addresses and their codes, available addressing modes and data sizes for various instructions, hand translation into machine language and determination of instruction timing, and a detailed discussion of the four basic status flags (the X flag is done in week 10) and how to derive the sixteen testable conditions from them. Crucial to this segment of the instruction is the use of a wide variety of tables, specially constructed for the course. A main component of the success of presenting this material, we feel, is the presence of ***one-page*** tables, each of which illustrates some aspect of the machine. These include a one-page machine language code table; a one-page timing table; a one-page table of all the missing addressing modes; a one-page table of register and flag capabilities; and several others.

The discussion of status flags is intended to bring out how sixteen tests, based on four flags, can give us, for both signed and unsigned numbers, the six relations (**u** < **v**, **u** ≤ **v**, etc.), used with `CMP`, and the six relations with zero (**u** < 0, **u** ≤ 0, etc.), used with `TST`, together with direct flag-set and flag-reset tests for each of the four flags. Included are a discussion of carry and borrow, and a proof of the fact that borrow is the complement of carry (if **u–v** is interpreted as **u+(–v)**). Also included is an introduction to the idea of using some of the conditional branches after `MOVE`, `SUB`, and the like (examples of this are given later). *Not* introduced yet are *Scc* (this is done in week 8, in connection with Boolean variables) or the use of the carry flag with shifting (this comes in week 7).

(6) The stack. Introducing all the properties of the 68000 hardware stack takes an entire week. The basic uses of the stack for return addresses, register saving and restoring, parameter passing, local variables in a procedure, and stack contents as an arbitrary mixture of data sizes, are discussed in a general setting. The hardware stack of the 68000 is then introduced, together with the stack pointer register and its uses with pre-decrement (pushing) and post-increment (popping). The various subroutine call instructions are introduced: `BSR`, `JSR` (together with `JMP`, which has been ignored until now); `TRAPV` and `CHK`; and A-traps and F-traps (`TRAP #k` is postponed to week 9, since it is not used on the Macintosh). `MOVEM`, both to and from memory, is discussed, as well as `RTS` (`RTE` and `RTR` are postponed to week 12). An ***elementary*** introduction to the four kinds of parameter passing (by reference, by value, by result, and by value and result) is given, with assembly language implications. The true specifications of `LINK` and `UNLK`, in stack frame allocation and deallocation, are given, as well as the explanation (finally!) of the procedure exit sequence from Chapter 2. Functional results are postponed to week 9.

At the same time, the students get their third programming assignment, which is an exercise in determining **var**- and non-**var**-parameters and performing the proper assembly language processing for each. It is a simultaneous determination of the maximum and minimum values in an array (adapted from [2]) which looks at array elements two at a time, comparing the smaller of the two (only) with the current maximum and the larger of the two (only) with the current minimum. (The result is a determination of both maximum and minimum in 3**n**/2 comparisons, rather than 2**n**.)

(7) Shifting, masking, and Boolean operations. Shifting and masking are normally taught at about this point (roughly halfway through an assembly language course). In our discussion of shifting, we postpone rotation-with-extend (and, indeed, all treatment of the X flag) until week 10. Treatment of arithmetic shifting, sign extension, logical shifting and rotation, masking and logical AND, and other logical operations is quite standard. Also treated are variable-length shifting, carry and overflow, immediate instructions to memory, and shifting and logical operations on words and long words. All these concepts are illustrated with examples. (A hint is given, at this point, as to an improvement of the solution to computer problem 3, due a week hence, by allowing the size of the array to be odd as well as even, and checking the rightmost bit of the length to determine whether it is even or odd.)

The presence of the sixteen *`Scc`* instructions on the Macintosh, as well as `BTST`, `BCLR`, `BSET`, and `BCHG`, requires us to give an unusually detailed discussion of Boolean variables. We start with a justification of keeping each unsubscripted Boolean variable in a byte rather than a single bit. We then pass to a discussion of Pascal statements of the form B := **true**, B := **false**, B := (U = V), B := C, B := B **and** C, and so on, for both unsubscripted Boolean variables and elements of packed Boolean arrays. Rotational shifting is illustrated by division of a packed array index by 8, using loading into the left half, clearing the right half, shifting right by 3, and then rotating the right half left by 3, thus moving three bits from the left-hand end to the right-hand end of the right half. The use of *Scc* instructions for B := (U = V) and the use of this in testing earlier values of variables ("if U *was* equal to V then go to Z") is expanded at this point to a discussion of saving and restoring the condition codes, which in turn leads to considering the CCR register (which has been completely ignored until now). In turn, `AND`, `OR`, and `EOR` to the CCR register show how flags can be set, reset, and changed.

(8) Characters and strings. We have postponed all but the most rudimentary discussion of character data until now in order to introduce a great variety of character-oriented notions all at once, together with a programming assignment on characters. The Macintosh extensions of ASCII are discussed, together with MPW Assembler character and string constants and the reasons for reserving zero as not the code for any character. MPW provides a built-in contrast between two character string conventions: MPW Pascal uses a length byte before each string, while the MPW C language uses a zero-byte ***after*** each string. These are contrasted by means of specific examples of string moving and comparison, one for each convention. Considerable time is devoted to the theory of string comparison, noting the similarities with arithmetic (unsigned) comparison and giving applications to sorting. A discussion of character-oriented functions (to test for the various kinds of characters, etc.) leads to a treatment of functions in assembly language, the `FUNC` and `ENDFUNC` directives, and functional results on the stack. Finally, there is an elementary treatment of input and output formatting: digit character code conversion by adding and subtracting `'0'`; the use of multiplication in input conversion, and division in output conversion; remainders on division and the `SWAP` instruction; and the handling of leading zeroes.

At this point the students get their fourth computer assignment, a combination of themes from weeks 7 and 8: they are asked to disassemble the machine-language register list in the `MOVEM` instruction. Since the bits in the register list are ordered either from left to right (for pushing with `MOVEM`) or from right to left (for popping with `MOVEM`), they need to use both left and right shifting, together with the carry flag. They are also asked to code an alternative version, using BTST, in which the tested bit is in a D-register to which 1 is added or subtracted. The output is an assembly-language-style register list (such as D3/A2–A4), given as a string with a length byte that must be calculated and stored; and the question of when to use the hyphen (–) in the register list requires logic which they are required to design themselves.

(9) Records and pointers. This material provides a transition between the treatment of these subjects received in an introductory Pascal course and the expanded treatment which the students will receive at a later time in a data structures course. The crucial idea here is that of the utility of the record concept; students should be encouraged to treat elements of records in the same way as other variables, with no loss of efficiency (in contrast to other data structures such as linked lists, whose efficient processing is beginning-to-end only). Similarities between array and record implementations (adjacency of addresses, A-register usage for pointers to arrays and records, and easy accessibility of components) are stressed at the outset. Absolute pointers (three bytes, usually kept in four) and relative pointers (two bytes) are compared and contrasted. Applications are given to commercial programming (a mailing list record), linked lists (a record consisting of content and pointer only), stack records (for parameters, temporary variables, and global variables), and static allocation, done by the MPW Linker in an area pointed to by A5. For linked lists, several assembly-language examples are given: searching, summation, and finally merging, a process in which linked lists save a considerable amount of space as well as a little time. Merging also gives us an introduction to symmetric algorithms, rôle switching (the smaller of the two current records having one rôle, the larger the other, with the two rôles represented by the two specific D-registers used as relative pointers) and the use of EXG in switching rôles (exchanging two D-registers in one instruction). Uses of pointers in jump tables, for implementation of the computed **go to** of FORTRAN and the **case** statement of Pascal (with applications to character processing in assemblers and compilers), are also given.

(10) **Large and long data and programs.** It is our strong belief that those features of assembly language which are useful only (or mainly) in *long* programs, or in the handling of large data (multiple precision integers, arrays of more than 64K bytes), *should not be introduced at all until near the end of the course.* Students have enough detail to learn, for their use in writing short programs. On the other hand, this material should not be skipped over entirely; it is important to understand the techniques required by large programs and data. These include the X flag and its application to double-precision and multiple-precision addition (with `ADDX`), subtraction (with `SUBX`), negation (with `NEGX`), comparison (with `CMPM`), and shifting by one bit (with `ROXL` and `ROXR`). General multiple-precision shifting, using masking, is also discussed, as well as the four dot-L addressing modes **k(An,Dm.L)**, **k(An,Am.L)**, **Q(Dn.L)**, and **Q(An.L)**. Also, BCD arithmetic, with `ABCD`, `SBCD`, and `NBCD`, together with general nybble (half-byte) processing and index calculations, are discussed, as well as the special treatment of the Z flag by `ADDX` and `SUBX`, allowing a test for multiple-precision zero. Also treated are assembler-language main programs, the `MAIN` and `ENDMAIN` directives, and further directives to improve the readability of long assembler listings (`TITLE`, `PRINT`, `EJECT`, `SPACE`, and `PAGESIZE`).

The remainder of this chapter is devoted to a discussion of macros and conditional assembly. At an earlier time, when we were teaching mainframe assembly language, we made a conscious decision, based on financial considerations, to postpone macros to a later course. The additional amount of money spent per assembly per student, if macros were used, was simply too much for us to justify. Using the Macintosh, this rationale disappears; assemblies remain longer with macros than without, but this does not result in additional outlays (we are not paying by the second any more). It is emphasized, however, that user-defined macros are useful mainly in large programs. Included here are the `MACRO` and `ENDM` statements; the macro prototype with its symbolic parameters; positional, keyword, and mixed-mode parameters, integer and string parameters; concatenation and the period operator; and macro nesting. There follows a section on conditional assembly and its expansion (using **go to** and SET-variables) into a full interpretive language interpreted by the macro processor in the assembler.

(11) **Real numbers and matrices.** We have postponed real numbers until now because the Macintosh does not come equipped with a floating-point coprocessor. In fact, the coprocessor does not even work with the 68000 that is in the Macintosh; it only works with the 68020, the 68030, and the like, so that one of these chips (available on an accelerator board) must be purchased first, and then the 68881 coprocessor chip added to this board. This roughly doubles the price of the basic Macintosh Plus. We are not concerned, in this week, with trying to introduce all the features of the 68881 (although most of them are covered in a subsequent appendix); we restrict ourselves to `FMOVE`, `FADD`, `FSUB`, `FMUL`, `FDIV`, `FNEG`, and `FMOD`, and use integer comparisons only. Beginning with a treatment of binary and hexadecimal fractions, floating point formats, biasing, normalization, zero (and the zero exponent as a code rather than an actual exponent), and fixed point data, we pass to considerations of the various instructions, registers, and data sizes, automatic data conversion, mixing of real and integer data and instructions (including the use of `FNOP` to synchronize the chips), parameter handling, machine directives, and the lack of volatile registers. This allows us to present the subtraction program of Chapter 2, recoded to use real numbers throughout.

The rest of this week is given over to a discussion of matrices. In working with subscripted variables in Chapter 4, we omitted two-dimensional arrays entirely, since there is quite a lot to be learned, and a programming assignment should be used here. The material here is oriented toward the presentation and explanation of an entire matrix-product subroutine which the students can copy and run. Row and column major, lower bounds of 0 and 1, arrays of bytes, words, and long words, square and rectangular matrices, and three-dimensional arrays are presented, and index calculation formulas for all of these given and presented in tabular form. We then pass to the relevant assembly-language techniques: brute-force index calculation; use of post-increment to process rows and of a generalized post-increment (using an actual `ADD`) to process columns; and the writing of a triple-loop program "from the inside out," writing the inner loop first, determining initialization values for it, then writing the middle loop, and so on. The students are now given their final exercise: writing a program similar to this, to transpose a real matrix.

(12) **System programs and input-output.** This week introduces the students to material upon which they will later expand, in a required undergraduate systems programming course. It is also a wrap-up of all those machine instructions and assembler directives which have not been presented so far. We start with absolute short addressing and low memory (with addresses $000000 through $007FFF). Certain locations in low memory turn out to be usable by user programs, such as the elapsed time since startup (at address $00016A), a pointer to the base of the stack (at address $000908), and the maximum RAM in this particular Macintosh (at address $000108). This leads to a discussion of the layout of addresses on the machine, with 4K reserved for RAM, the next 4K for ROM, and the final 8K for the various peripheral chips. We then pass to interrupts and exceptions; the exception processing routine addresses kept in the first 256 bytes of memory; `RTE`, `RTR`, and the general handling of status bits during exception processing; and other system-oriented instructions such as `RESET`, `MOVEP`, `TAS`, and `STOP`.

The basics of input-output are now taken up. This includes the general principles of memory-mapped input-output; the screen buffer (having a pointer at $000824 to its starting address) and the way it is scanned; the lack of built-in character-oriented output and the formation of characters on the screen as dot-matrix patterns; unified treatment of keyboard input and mouse movements as "events," and the system subroutine GetNextEvent, which returns either a keypress or a mouse movement; and a very elementary appreciation of disk file systems and disk-oriented special-purpose chips such as the IWM.

(13) **Further topics.** In the last week of the course, we present a wealth of new material, for which the student is not responsible at this time, but which should help in transition to later courses. This is presented in the form of further handouts, about the size of those which the students have been getting so far. Two such handouts have already been given; one, in week 4, describes the debugger (currently MacsBug) which the students may use to help debug their programs, while the other, in week 11, gives further information on the 68881. The new handouts are as follows. There is one on C, done from an assembly language perspective. (I believe that people writing C programs should consider, during the programming process, how these programs would be written in assembly language. This should produce much more efficient C programs, which are at the same time reasonably close to being portable.) There is one on the 68020, which has a few dozen extra instructions. There is one on the 6502, to give the contrast between 16-bit and 8-bit architectures. There is one which completes our discussion of macros and conditional assembly, with several further examples. Finally, there is one on graphics programming in general, and on the Macintosh system subroutines in particular (that is to say, the subroutine package known as Quickdraw).

All this is accompanied by tables, exercises, answers to half the exercises, a short bibliography, and a long and complete index. There is actually enough material here for a full year's worth of study, although we do not do this because we consider it more important, in the second semester, for the students to consider a different assembly language, such as that of the Intel 8086 family. In addition to the introductory assembly language class described above, this material has also been used in a graduate course, entitled Languages for System Programming, which covers both C and 68000 assembly language. Since these students have all had some other assembly language, as well as Pascal, we are able to cover the material at double speed, employing one week for every two weeks' worth of work described above. Thus we are able to finish 68000 assembly language in the first half of this graduate course, allowing the rest of it to be devoted to C.

References

1. Maurer, W. D., *Apple Assembly Language*, Computer Science Press, Rockville, MD, 1983.

2. Horowitz, E., and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Rockville, MD, 1983, pp. 108-112 and problem 11, pp. 142-143.

PROGRAMMING EFFORT AND PROGRAM QUALITY: QUANTIFYING THE RELATIONSHIPS

Barry J. Donahue
Computer Science Department
Central Washington University
(509) 963-1495

Abstract

The purpose of this study was to investigate several aspects of the programming task. Of particular interest were the relationships between program quality and both the effort expended in the writing of the program and the distribution of programming effort over the available time. Also of interest were the perceptions of the programmers concerning the amount of effort they put forth. The results of the study suggest some interesting conclusions. (1) Programmers tended to underestimate the number of program runs they had made. This was a fairly consistent underreporting, however, and could be predicted quite accurately. (2) Once the programmers began working on the program, they tended to work until the assignment was due. Thus, those who began work promptly generally did not finish before their more tardy peers. (3) Most surprisingly, however, the quality of a subject's program was not related to the quantity of effort expended. Further, a subject's knowledge of a new language was not related to the quantity of programming effort. Thus, the amount of time a programmer spent in writing the program bore little relationship to how good a program he wrote or how well he had actually learned the new language.

When evaluating software, most of the criteria can be reduced to two key factors: the quality of the finished product and the amount of time required to produce it. Much research has been undertaken to quantify each of these aspects of software design [1] [2]. Unfortunately, little has been done in a consistent, quantitative fashion to explore the relationship between these two factors. There have of course, been numerous anecdotal and descriptive accounts of this relationship [3] [4]. Such accounts have been very helpful in giving us a certain intuitive feel for the complex interactions that take place between qualitative and temporal factors. Still, it is of great importance to begin to apply a more rigorous analysis to this problem in an effort to clarify for software designers the degree to which these factors influence one another.

This desire for clarification was the primary motivation for undertaking the present study. There were, however, other questions of interest which needed to be addressed as well. Foremost among these was an investigation of the ability of programmers to estimate, after the fact, the amount of effort they had expended on the development of a piece of software. Again, this is a question that has received very little attention. Most of the work in this area has focused on the accuracy of prediction; that is, how accurate are pre-design man-hour estimates of the length of the design life cycle [5].

Another area in which definitive answers are as yet unavailable is that of determining how the programming effort is distributed over the available on-task time. Is it best to work relatively harder at the beginning of a project or at the end? Or should effort be balanced over the entire course of development? An important question to answer would be whether certain patterns of distribution of effort lead to better or more timely software.

Finally, a question of particular importance to educators concerns the correlation between the quality of a program and a written examination of a student based upon the syntax and semantics of the language used in the program. In courses designed to teach a particular language, say Pascal, it would be comforting to know that programming assignments and tests about Pascal have a high degree of consistency and that scores on each correlate well. The students in the course CS 361: Programming Language Theory were the subjects for the study. All of these students had considerable programming experience in Pascal, FORTRAN, COBOL, and Assembly languages. Most had also had courses in data structures and operating systems, as well as other advanced course work.

In the Programming Language Theory course, students study the principles of programming languages and become conversant with two or three special purpose languages such as LISP, SNOBOL, C, or APL. The first language to which the students were introduced in the course was SNOBOL. After two weeks of instruction, they were given a simple programming assignment. They had 9 days to complete the assignment. The SNOBOL interpreter generates a log of the account name and time of access for all accounts which access it. This was the only SNOBOL interpreter to which students had access--they could not practice on another system. After the assignment was collected, this accounting file was sorted into a file listing each account and the number of accesses that account made to the interpreter per day. When they had turned in their programs, the programmers were queried as to their perceptions of the language and were also asked to estimate the number of times they had executed their programs. They had not known this information would be requested of them.

These data were used to describe the programmers' efforts in completing their assigned task. There were four aspects of effort which were measured: effort expended, distribution of effort, quality of effort, and estimation of effort. Various statistical techniques were used to produce and analyze these four measures and the relationships between them. The measures are discussed below and are shown in Figure 1.

A) EFFORT EXPENDED
  1) Total Number of Runs
  2) Mean Runs: Total Runs/Days Available
  3) Number of Days Worked
  4) Adjusted Mean Runs:
     Total Runs/Days Worked

B) DISTRIBUTION OF EFFORT
  1) Mean Work Day: "Average Day Worked"
  2) Day Started
  3) Standard Deviation of Runs
  4) Coefficient of Variation:
     Standard Deviation/Mean Runs

C) QUALITY OF EFFORT
  1) Program Grade
  2) Test Score

D) ESTIMATES OF EFFORT
  1) Estimated Runs
  2) Estimation Ratio:
     Estimated Runs/Actual Runs
  3) Estimation Relative Error:
     1 - Estimation Ratio

Figure 1. Summary of metrics used in the study

The amount of effort expended was measured in four ways. First, the total numberof executions was determined from the log. This number was divided by the number of days available to give the mean number of runs per day. (In most cases these two values will produce the same results when manipulated statistically. However, as it is sometimes more meaningful to talk about one or the other, both were used.) Next, the number of days on which a programmer worked was computed from the log. Finally, it was felt that a more accurate average of runs might be that obtained by considering only the days on which a programmer worked. This was computed to give the adjusted mean runs.

Distribution of effort was also measured in four ways. The first of these was designed to provide an estimate of central tendency in the distribution of a programmer's work. This mean work day for a programmer is defined by:

$$w = \frac{\sum_{i=1}^{9} ir_i}{t}$$

where r is the number of program executions by a programmer on the ith day and t is the sum of program executions for a programmer.

Thus, the higher the value of w for a programmer, the later in the period his "average" day of work would be. The second measure of distribution of effort was simply the ordinal number of the day on which a programmer began work. A value of 4 would indicate the programmer waited until the fourth day to execute his program for the first time. The standard deviation of daily program executions was the third measure of distribution of effort, and the coefficient of variation provided the final measure. The coefficient of variation tends to correct for standard deviations that are large or small simply because the number of runs is relatively large or small. Thus it makes standard deviations for programmers with different numbers of runs more comparable.

Quality of effort was measured in two ways. The principal measure was an evaluation of the resulting program by the researcher. Values of 0 to 100 were possible. The second measure was the score on a test of syntax and semantics (also a range of 0 to 100) given to the programmers about one week after the due date of the assignment.

In order to answer questions concerning a programmer's ability to estimate the amount of effort expended, the subjects were asked to give an estimate of the number of program executions they had made during the development process. Three measures of estimation of effort were used in the study. The raw estimate was the first of these measures. Next, a ratio of the estimated number of runs to the actual number of runs was generated. This ratio (e) proved difficult to interpret statistically in certain cases; it indicates whether an estimate is above (e > 1) or below (e < 1) the actual number of runs. When we are concerned only with the amount of error, this directional characteristic becomes problematic. As a result, the Estimation Relative Error was used to indicate only degree of error. It is defined as

$$| 1 - e |$$

The larger the value of Relative Error, the worse was the programmer's estimate.

RESULTS

Table 1 shows the raw data and also the values for the various measures of effort expended. The intersection of the third row and column 6, for example, tells us that on the 6th day programmer four made 29 runs of his program. The programmers averaged about 91 total runs at about 10 runs per day. As can be seen, the ranges are considerable. About 3/5 of the available days were used. The average of runs on days worked was roughly 16.

The data corresponding to the measures of distribution of effort appear in Table 2. The average day started was roughly 1/3 of the way into the available time period, while the mean work day average of 5.6 suggests the average day of work was at roughly the 2/3 mark through the period. The coefficient of variation is substantially less volatile than the standard deviation, suggesting that some of the variation in the average number of runs is due simply to the variation in total number of runs.

The measures of quality of effort and the measures of estimation of effort are shown in Table 3. The actual number of program executions is also shown for comparison purposes. Note that all but one of the estimates are lower than the actual number, and many are substantially lower. The mean of the estimation ratio is .48. In other words, the programmers on average estimated that they had executed their programs only 1/2 as many times as they actually had executed them, even though the task had been completed only days before. While the Estimation Ratio gives a good measure of the direction of error in estimation, the Estimation Relative Error provides information on the amount of error, irrespective of direction. It gives the percent of misjudgment of the estimate. This measure shows that the programmers tended to be about 45% in error of the actual number of program executions.

A level of significance of 0.05 was chosen for the study. It was felt that such a level would be demanding enough to exclude chance relationships from appearing as significant and yet would enable the researcher to detect significant relationships that might be excluded by a smaller level of significance. When the tests of significance were performed, however, it became obvious that the important results settled nicely into either the significant or the insignificant categories, there are no borderline cases. This was somewhat surprising as it was not expected that an initial study would provide so many conclusive results.

There were few surprises among the significant results. These results are given in Table 4. A T-test was performed between the actual number of runs and the estimated number of runs. The 2-tailed T value of -4.05 is significant, confirming he assumption that there is a statistical difference between the actual number of program executions and estimates of this number. When these two sets of values are tested for correlation (Pearson r), however, a significant correlation is found. This indicates that, while postmortem estimates are generally lower than the actual number of executions, these estimates tend to be roughly the same fraction of the actual number of runs for any particular programmer. This correlation and the resulting regression line may be seen in Figure 2.

There is a significant negative correlation between the day a programmer began work and the number of days worked. This, together with the negative correlation between mean work day and the number of days worked, suggests that the programmers tended to use all of the available days no matter when they began. If this were not the case, we could expect no significant correlation of these two factors. Further support for this conclusion may be found in the significant correlation between total number of runs and number of days worked. The negative correlation between total number of runs and mean work day simply suggests that those who started late did not have time to run their programs as often as those who started earlier.

The significant correlations between day started and the coefficient of variation and between number of days worked and the coefficient of variation (note this correlation is negative) indicate that there is less variation in runs per day as the number of days worked increases. Thus, the earlier in the period in which a programmer began work, the more consistent he tended to be in his work pattern. This conclusion may need some tempering, however, as it should be realized that values of 0 runs on those days on which a programmer did not work may have had a large impact on the standard deviation and hence the coefficient of variation. This result,therefore, could probably admit of further, more controlled testing.

As we might hope, there is a significant correlation between the score a program was assigned when evaluated and the score received by the programmer on a test of language syntax and semantics. This may suggest to educators that they can safely utilize the results of a well-designed test given in a well-controlled environment as a valid indicator of programming skill. This would allow the educator to assign a relatively smaller percentage of total points in a course to programming grades.

The study produced two quite surprising significant correlations. Separately, it might be tempting to dismiss them as arising by chance; but, as they reinforce one another, the results must be considered carefully. The scores on the test the programmers were given correlated significantly with both the day started and the coefficient of variation. The first of these indicates that the test score tended to be higher if the programmer began work later. The second indicates that a programmer's test score tended to be higher the more variable the distribution of effort. The natural assumption that programmers who began early and worked roughly the same amount each day would have had a more complete knowledge of the programming language being learned appears not to be the case. We must be careful not to attribute any causality to this relationship. Nonetheless, the results are provocative.

The final significant correlation of interest occurs between the Estimation Relative Error and mean work day. This negative correlation suggests that programmers tended to give better estimates of the amount of work they had performed as the bulk of that work was done later in the period.

By far the most interesting aspect of the study concerns those results which did not prove to be significant. Many of these results counter our intuitive notions of the principles of software design. In all cases the probability p was larger than 0.10. These results are summarized in Table 5. Most perplexing of all are the results related to program grade. None of the measures of effort expended correlate with the quality of effort as measured by program grade. This would seem to indicate that the amount of effort which a programmer spends bears no relationship to the quality of his product. Some programmers were able to produce quality programs with little apparent effort; others required much time, but also produced good programs. Poor programs were produced by both types of programmers, as well.

Also surprisingly, none of the measures of distribution of effort correlate with the program grade. Our notions of the advantages of adherence to a consistent approach to the programming task (if, indeed, we hold such notions) may be suspect.

Similar results obtain with respect to test scores. None of the measures of effort expended correlate significantly with test score. Apparently, the amount of effort programmers expended on their programming project had no relationship to the score they received when tested on the syntax and semantics of the language. The situation is not quite so clear when test score and the measures of distribution of effort are examined. Two of the resulting correlations are significant and two are not. Until further research clarifies this situation, the assumption should be made that here, too, relationships are weak or non-existent.

The lack of significant correlation between effort expended and both program grade and test score is, as has been noted, somewhat surprising. On first examination it would probably be assumed that the more time a programmer took while developing a program, the more experience with and hence knowledge of the programming language being learned would increase. But upon further reflection, it would seem that some programmers simply learn a language faster than others and no qualitative descriptors can be attached to this knowledge. Perhaps "feel comfortable with" should be read here in place of "learn." Note that the correlation is NOT negative; i.e., it is not the case that those who program faster learn a language to a qualitatively superior degree. Thus, it appears to be very necessary to separate the effort expended on a project and the quality of the project; there is simply no correlation--positive or negative--between the two.

Delimitations and Suggestions for Further Research

As has been noted, the results of the study suggest many interesting conclusions. Still, it is important to recognize some of the factors which may have contributed to the results but which were not readily measurable. The most important of these is undoubtedly the time the subjects spent in design and desk checking. The present study dealt only with the coding aspect of program generation. Perhaps, for example, those students who made few runs but produced high quality programs spent far longer in the design and walkthrough phases than did their peers. As it is very difficult to measure these aspects of programming as undertaken by students, similar studies which focus on pre-coding activities should be undertaken.

The best environment for such studies would probably be a large software engineering project. In such an environment it would be easier to monitor on-task design time. Also, as the time frame is much longer, it would be possible to break the project into several relatively discrete phases.

A second major delimitation of the study concerns the measures of quality. It may be wise to separate several factors out of the overall judgment of quality and rate each of these factors individually. Thus, programs may be judged on factors typically given as characteristics of program quality such as those given by McCall et al [6]:

usability
integrity
efficiency
correctness
reliability
maintainability
testability
flexibility
reusability
portability
interoperability

Implications

The results of the study suggest that certain behaviors and expectations on the part of those managing programmers and software projects should be modified. First,in learning situations, instructors need to be aware that students are not particularly good judges of how much work they have done. If we generalize these results to industry, we may find a good explanation for a question which has perplexed software design managers since the first large software products were produced: Why do software projects invariably run behind schedule? One reason may very likely stem from this inability to judge correctly our past efforts, no matter how recent. A programmer will tend to underestimate the time required to complete a new project simply because he does not remember accurately how much time it took him to complete a similar project in the past. Thus, the old maxim: "Estimate as carefully as you can the amount of time you think the project will take--and then double it" seems more nearly to capture the truth of estimation error than do more esoteric explanations.

Secondly, we should be careful not to judge our programmers—whether students or employees—on how much effort they seem to be giving to the task. This is not to suggest, of course, that the employee whose day is simply an eight hour coffee break ought not to arouse suspicions, but it is merely a caution that the programmer who expends the most effort according to the most easily observable behaviors might--or might not--produce the best result. Thus, enforcing rules for generating a specific number of lines of code, number of executions, or other measures of productivity is probably counterproductive. It may raise the level of stress in the organization but will do little to raise the level of quality.

Similarly, it is probably counterproductive to attempt to force an even distribution of end product generation. Managers need to supply the environment and the tools; programmers must produce the code.

Finally, since good programmers are at a premium, it would seem to be most cost-effective to retain those who produce quality results, realizing that there may be considerable variance in the amount of time required to produce those results. Studies show that maintenance costs account for approximately 70% of the cost of a software system [7]. Since maintenance is so expensive, it should be efficient to recognize those who are able to save as much of this cost as possible by initial quality workmanship. Programmers who are fast but who produce inefficient or error-prone code should be expendable. It is surprising how many software producers fail to recognize this fact. It is a key reason why, despite the development of principles of software engineering such as modularity and structured design, the "software crisis" remains with us.

Conclusions

The following conclusions are drawn from the results of the study.

A) The subjects tended to begin work about the 3rd day out of the 9 available days.
B) The subjects tended to be half finished about the 6th day of the 9 available days.
C) The subjects tended to underestimate the number of program runs to a considerable degree.
D) The subjects tended to use all of the time from the day they began work until the assignment was due.
E) The subjects tended to distribute work more evenly across the available days the earlier they began work.
F) The quality of programs was not related to the quantity of effort.
G) The quality of programs was not related to the distribution of effort.
H) The quality of knowledge concerning the new language was not related to the quantity of programming effort.
I) The quality of knowledge concerning the new language was generally not related to the distribution of effort.

References

1. B. W. Boehm, *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.
2. C. E. Walston and C.P. Felix, "A Method of Programming Measurement and Estimation", *IBM Systems Journal*, 16 (1), 54-73 (Jan 1977).
3. F. P. Brooks, *The Mythical Man-Month*, Addison-Wesley, Reading, Massachusetts, 1975.
4. E. W. Dijkstra, "The Humble Programmer", *Communications of the Association of Computing Machinery*, 15, 859-886 (Oct 1972).
5. S. N. Mohanty, "Software Cost Estimation: Present and Future". *Software--Practice and Experience*, 11 (2), 103-121 (Feb 1981).
6. J. A. McCall, P.K. Richards, and G.F. Walters, "Factors in Software Quality", General Electric, Command and Information Systems, (Technical Report 77CISO2), Sunnyvale, California, 1977.
7. M. M. Lehman, "Programs, Life Cycles and the Laws of Software Evolution", *Proceeding IEEE*, 68 (9), 1060-1076 (September 1980).

	RUNS PER DAY									TOTAL RUNS	MEAN RUNS	DAYS WORKED	ADJUSTED MEAN
	1	2	3	4	5	6	7	8	9				
1	0	10	52	7	6	2	0	1	2	80	8.9	7	11.4
2	26	17	2	3	0	0	7	5	0	60	6.7	6	10.0
3	0	15	23	25	46	29	0	0	17	155	17.2	6	25.8
4	0	0	12	3	27	4	0	0	7	53	5.9	5	10.6
P 5	0	0	0	0	0	3	27	24	8	62	6.9	4	15.5
R 6	0	0	48	10	0	8	4	10	0	80	8.9	5	16.0
O 7	25	47	63	15	46	59	1	19	2	277	30.8	9	30.8
G 8	0	1	4	0	0	4	3	4	5	21	2.3	6	3.5
R 9	0	0	0	0	0	0	0	11	0	11	1.2	1	11.0
A 10	21	1	32	0	61	41	0	0	53	209	23.2	6	34.8
M 11	0	0	0	0	16	0	16	0	15	47	5.2	3	15.7
M 12	1	0	35	22	0	13	0	12	17	100	11.1	6	16.7
E 13	0	14	12	0	7	9	0	38	1	81	9.0	6	13.5
R 14	0	0	0	8	0	21	27	21	7	84	9.3	5	16.8
15	8	3	0	6	15	9	0	0	20	61	6.8	6	10.2
16	0	13	42	48	5	16	0	6	3	133	14.8	7	19.0
17	0	0	6	0	0	18	0	2	1	27	3.0	4	6.8
18	0	0	0	3	0	0	0	6	15	24	2.7	3	8.0
19	0	0	6	0	19	98	0	35	2	160	17.8	5	32.0
SUM	81	121	337	150	248	334	85	194	175	1725			
MEAN	4	6	18	8	13	18	4	10	9	91	10.1	5.3	16.2

Table 1. Effort Expended

		RUNS PER DAY									TOTAL RUNS	MEAN RUNS	STD DEV	MEAN WORK DAY	DAY STARTED	COEF OF VAR
		1	2	3	4	5	6	7	8	9						
	1	0	10	52	7	6	2	0	1	2	80	8.9	16.5	3.4	2	1.86
	2	26	17	2	3	0	0	7	5	0	60	6.7	9.0	2.8	1	1.35
	3	0	15	23	25	46	29	0	0	17	155	17.2	15.6	4.9	2	0.91
	4	0	0	12	3	27	4	0	0	7	53	5.9	8.9	5.1	3	1.51
P	5	0	0	0	0	0	3	27	24	8	62	6.9	10.9	7.6	6	1.58
R	6	0	0	48	10	0	8	4	10	0	80	8.9	15.3	4.3	3	1.72
O	7	25	47	63	15	46	59	1	19	2	277	30.8	23.6	4.1	1	0.77
G	8	0	1	4	0	0	4	3	4	5	21	2.3	2.1	6.5	2	0.88
R	9	0	0	0	0	0	0	0	11	0	11	1.2	3.7	8.0	8	3.00
A	10	21	1	32	0	61	41	0	0	53	209	23.2	24.5	5.5	1	1.06
M	11	0	0	0	0	16	0	16	0	15	47	5.2	7.8	7.0	5	1.50
M	12	1	0	35	22	0	13	0	12	17	100	11.1	12.3	5.2	1	1.10
E	13	0	14	12	0	7	9	0	38	1	81	9.0	12.2	5.8	2	1.35
R	14	0	0	0	8	0	21	27	21	7	84	9.3	10.8	6.9	4	1.16
	15	8	3	0	6	15	9	0	0	20	61	6.8	7.1	5.7	1	1.05
	16	0	13	42	48	5	16	0	6	3	133	14.8	18.0	4.1	2	1.22
	17	0	0	6	0	0	18	0	2	1	27	3.0	6.0	5.6	3	1.99
	18	0	0	0	3	0	0	0	6	15	24	2.7	5.1	8.1	4	1.90
	19	0	0	6	0	19	98	0	35	2	160	17.8	32.4	6.2	3	1.82
MEAN											91	10.1	12.7	5.6	2.8	1.46

Table 2. Distribution of Effort

	PROGRAM GRADE	TEST SCORE	ACTUAL RUNS	ESTIMATED RUNS	ESTIMATION RATIO	ESTIMATION REL ERROR
1	100	18	80	10	.13	.88
2	100	15	60	15	.25	.75
3	100	11	155	25	.16	.84
4	73	17	53	--	--	--
5	98	18	62	15	.24	.76
6	93	15	80	20	.25	.75
7	85	18	277	70	.25	.75
8	20	9	21	17.5	.83	.17
9	93	19	11	3.5	.32	.68
10	73	11	209	60	.29	.71
11	98	20	47	45	.96	.04
12	98	12	100	20	.20	.80
13	98	13	81	60	.74	.26
14	98	18	84	80	.95	.05
15	98	13	61	50	.82	.18
16	98	15	133	75	.56	.44
17	100	24	27	6	.22	.78
18	93	12	24	30	1.25	.25
19	70	10	160	30	.19	.81
MEAN	88.7	15.2	91		.48	.55

Table 3. Quality of Effort and Estimation of Effort

1)	ESTIMATED NUMBER OF RUNS with ACTUAL NUMBER OF RUNS	T = -4.05	p = .001 (2-tail)
2)	ESTIMATED NUMBER OF RUNS with TOTAL NUMBER OF RUNS	r = 0.5408	p = .010
3)	DAY STARTED with NUMBER OF DAYS WORKED	r = -0.8543	p = .000
4)	MEAN WORK DAY with NUMBER OF DAYS WORKED	r = -0.7502	p = .000
5)	TOTAL NUMBER OF RUNS with NUMBER OF DAYS WORKED	r = 0.6835	p = .001
6)	TOTAL NUMBER OF RUNS with MEAN WORK DAY	r = -0.4124	p = .040
7)	DAY STARTED with COEFFICIENT OF VARIATION	r = 0.7429	p = .000
8)	NUMBER OF DAYS WORKED with COEFFICIENT OF VARIATION	r = -0.7653	p = .000
9)	PROGRAM GRADE with TEST SCORE	r = 0.4781	p = .019
10)	DAY STARTED with TEST SCORE	r = 0.4495	p = .027
11)	COEFFICIENT OF VARIATION with TEST SCORE	r = 0.4314	p = .033
12)	ESTIMATION RELATIVE ERROR with MEAN WORK DAY	r = -0.4518	p = .030

Table 4. Significant Results ($p < 0.05$)

1) PROGRAM GRADE

Measures of Effort Expended

TOTAL NUMBER OF RUNS	p = .485
MEAN RUNS	p = .485
NUMBER OF DAYS WORKED	p = .314
ADJUSTED MEAN RUNS	p = .457

Measures of Distribution of Effort

MEAN WORK DAY	p = .269
DAY STARTED	p = .293
STANDARD DEVIATION OF RUNS	p = .498
COEFFICIENT OF VARIATION	p = .172

2) TEST SCORE

Measures of Effort Expended

TOTAL NUMBER OF RUNS	p = .162
MEAN RUNS	p = .162
NUMBER OF DAYS WORKED	p = .117
ADJUSTED MEAN RUNS	p = .136

Some Measures of Distribution of Effort

MEAN WORK DAY	p = .472
STANDARD DEVIATION OF RUNS	p = .127

3) ESTIMATION RELATIVE ERROR

DAY STARTED	p = .312
COEFFICIENT OF VARIATION	p = .201
NUMBER OF DAYS WORKED	p = .231

Table 5. Insignificant Results ($p > 0.10$)

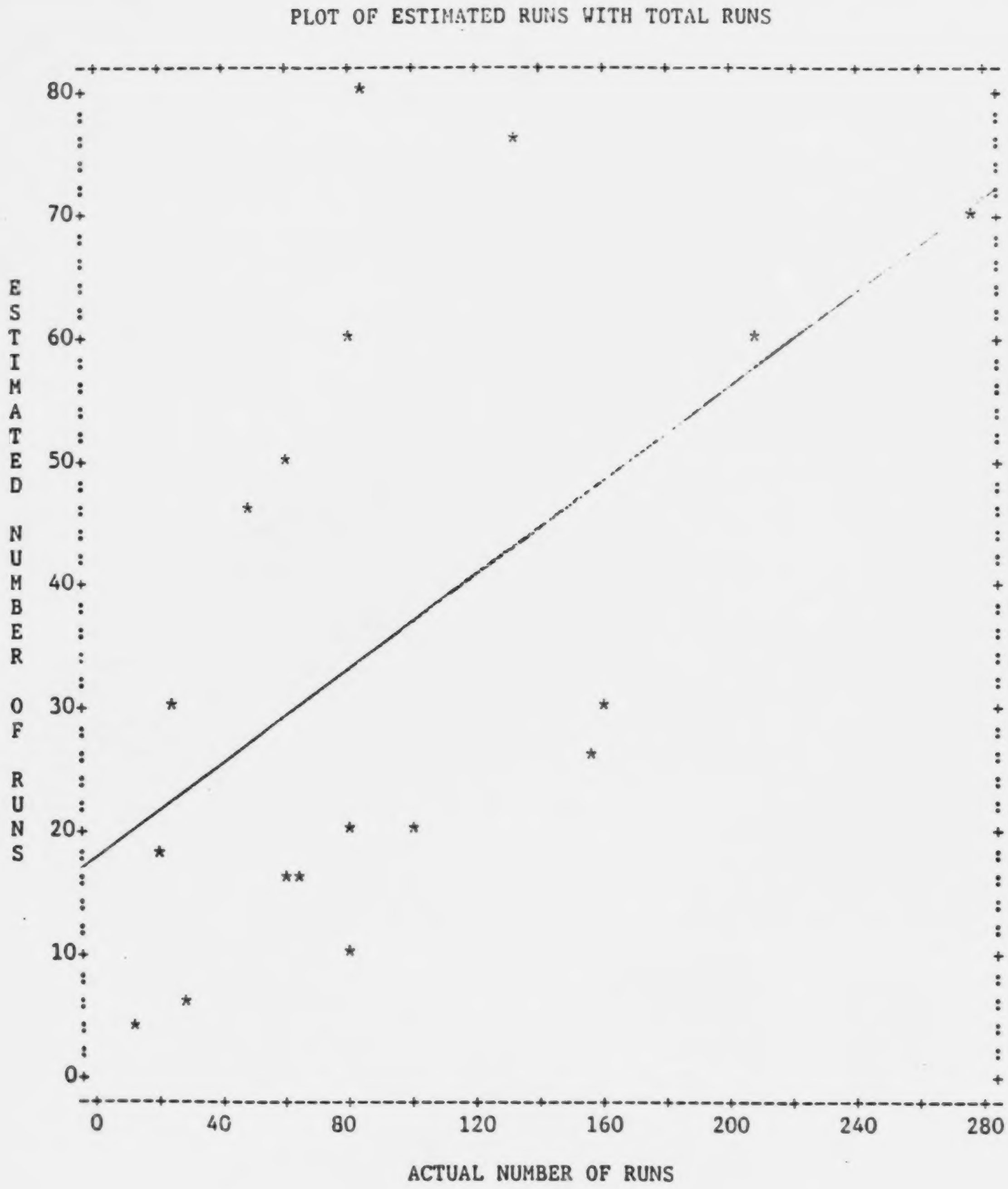


Figure 2. Regression of Estimated Number of Runs with Actual Number of Runs

INTERACTIVE COACHING FOR CURRICULUM-BASED INFORMATION TOOLS

Charles S. White
Center for Interactive Educational Technology
George Mason University
4400 University Avenue
Fairfax, VA 22030

Jos. J. Beishuizen
Frances M.T. Brazier
Vrije University
The Netherlands

Beverly Hunter
Targeted Learning Corporation
Amissville, VA

Jeanne C. Meadows
Cooper Intermediate School
McLean, VA

ABSTRACT

There is increasing interest in the application of large databases, multimedia systems, hypertext, and other information systems to learning and teaching. However, we know little about how teachers and students acquire the skills needed to make productive use of such tools and information in an educational "overlay" on tool software, performing as an intelligent coach or advisor. In this session, we explore examples of this work being conducted in the United States and in the Netherlands.

The former is represented by the *Scientists At Work* project, which produced and tested a computer program to help middle and high school children acquire and apply science inquiry skills within life science content. *Scientists At Work* is a HyperCard-based program that includes seven main components:

- a *database* containing pictures, text, numeric and sound information on 170 animal species and their habitats;
- software *tools* for sorting, viewing, arranging, selecting, graphing and printing the data;
- an interactive coach that provides help in learning and applying science inquiry skills and informating-processing skills;
- five proposed *projects* relevant to the middle school life science curriculum;
- a *dictionary;*
- a *student workspace* for taking notes; and
- *student records* to help the teacher in monitoring and assessing students' progress.

Beverly Hunter will discuss design challenges associated with the project, and Jeanne Meadows, one of the field-test teachers, will describe students' use of*Scientists At Work* in a classroom setting. Charles White will present the results of a study conducted during the field test to gauge the effectiveness of the product with respect to selected information-processing skills and student attitudes, and will discuss the broader research issues that emerged from the project's evaluation phase.

Illuminating key research and development issues surrounding interactive coaching is the work of Frances M.T. Brazier and Jos J. Beishuizen. Both have conducted research on interactive coaching systems and information-retrieval interfaces grounded in cognitive psychology. Brazier and Beishuizen have explored differences in information retrieval ability between a menu-driven versus a command-driven database system. Beishuizen examined directly the use of interactive coaching in information retrieval. He found positive gains in retrieval efficiency using the coach, especially when students struggled with weakly-structured data. The presenters from the Netherlands will describe their current work in the design and testing of interactive coaching systems.

COMPUTING IN CAREERS/CAREERS IN COMPUTING
A KIT FOR TEACHERS, COUNSELORS, AND VOLUNTEERS

John Lawson, Chair
Chair, ACM/Careers Project
Lewis Clark State College
Lewiston, ID

ABSTRACT

Knowledge of the training requirements, salary expectations, and potential for advancement in a wide variety of computer careers should be made available to all students at the junior and senior high school levels. There is evidence that carefully planned, well integrated career counseling helps students make more realistic career choices, succeed in completing their preparation, and accept positions in their chosen career.

The Education Board of the Association for Computing Machinery (ACM) commissioned a seven-member task force to develop a plan to deliver current, accurate information on computing in careers and careers in computing to secondary students. The task force has produced a kit of materials and activities for use by teachers, counselors, or computer industry volunteers with students in grades 7-12.

There are three primary objectives for the Careers in Computing project:

- to provide a comprehensive package of current, usable information on careers which utilize computers daily;
- to provide teachers and counselors with means of locating and using resources in technical career guidance;
- to encourage greater involvement by representatives from the computer world outside of schools.

The kit materials consist of a videotape, class sets of several brochures, bibliography of resources, copy masters for presenters, and a presenters' manual with classroom activities. Plans call for production of a training videotape to be released in March 1989 which will describe the design philosophy of the kit, showing presenters using each of the activities with typical classrooms. Brochures included in the kit contain pictures, job responsibilities, training requirements and brief interviews with typical workers in a variety of job categories.

The Presenters' Manual is a loose-leaf guide to suggested uses of the kit. It contains an inventory list, hints for successful presentations, 12 suggested activities for students, the bibliography of related materials, and a form for recording the number of uses of the kit.

Plans call for distribution of a minimum of 100 kits per year in the 88-89 and 89-90 school years. Distribution is based on geographical diversity, type of school, and commitment to use the kits as demonstrated by letters of support from school representatives.

The first round of distribution and results from actual classroom use will be presented. The session will also introduce the materials in the kit, review the training video, discuss evaluation, and inform participants of how to obtain a kit.

PANELISTS

Doris Carey
University of Colorado at Colorado Springs
Colorado Springs, CO

Bill Baird
Auburn University
Auburn, AL

Teacher or Guidance Counselor
who has actively used the Kit
To be announced

SPONSOR: SIGCUE

ARTIFICIAL INTELLIGENCE AND COMPUTER SCIENCE

Michael R. Muir
Susan N. Kirkpatrick, Barbara Biglan
Ted Mims, Michael Jones

ABSTRACT: The Fantastic Journey Project

Michael R. Muir, Burnham, ME 04922.

The Fantastic Journey Project is a high school Gifted and Talented course exploring artificial intelligence and Logo. It was offered as a Regional Gifted and Talented program for students in Waterville, Winslow, Oakland, and Fairfield, Maine, for the 1988-1989 school year. The project is sponsored by an Innovation Education Grant and State Competitive Grant for Gifted and Talented Education from the Maine Department of Education and the1988 UVA Summer Logo Fellowship Program from LCSI and the University of Virginia.

Three students from each school met for the whole day, two times a month, at Winslow High School's computer lab. There, they shared and created artificial intelligence programs. Students participated in debates, reported on independent research, and designed their own humanistic programs. Each chapter of the text, which was developed and written by the instructor, introduces a new AI concept and a program illustrating the concept. The text includes ideas for improvements and extensions to the program and students were encouraged to implement them. Students shared their programs and revised them much like students do in the Writing Process. This open-ended cycle facilitates critical and creative thinking and problem solving as well as the learning of new facts and ideas.

The objectives of the program include challenging and motivating gifted high school students and serving both as a model for a new approach to computer science and for advanced uses of the language Logo.

ABSTRACT: Integrating Short AI Programs in Logo and BASIC into Content Areas

Susan N. Kirkpatrick, Shady Side Academy Middle School, 500 Squaw Run Road East, Pittsburgh, PA 15238; Barbara Biglan, Shady Side Academy Junior School, Pittsburgh, PA 15221.

This presentation will give an overview of the use of artificial intelligence in education at Shady Side Academy. It will relate ways in which this highly complex topic can be watered down for secondary and elementary classooms. Short AI programs in Logo and BASIC will be demonstrated, and suggestions will be given for implementing their use in content areas.

There are four parts to Shady Side Academy's AI curriculum. Part 1 is a general introduction to AI. Activities explore questions such as, What is intelligence?, What is artificial intelligence?, How do computers learn?. Part 2 deals with AI applications. Students look at a variety of educational AI programs that are easily obtainable, many of them being public domain. The different types of AI programs that are studied include: games and simulations, expert systems, problem solving, and natural language processing programs. Students research and discuss the natural language processing programs. Students research and discuss the use of AI in business, industry, medicine, and law. At local universities students actually use intelligent tutoring systems (ICAI). Part 3 deals with the programming aspects of AI programs. All students must have knowledge of programming in Logo and BASIC, and advanced students will be introduced to Prolog or Lisp. The goal here is not to create an elaborate AI program, but to explore its structures. The differences between algorithmic and heuristic programming are explored in this part. In Part 4 students build robots, some of which can be interfaced with a computer and programmed to do simple tasks. The relationship between robots and AI is discussed, as well as language processing, visual processing, and locomotion of robots.

ABSTRACT: Microcode Simulator

Ted Mims, Michael Jones, Computer Science Department, Nicholls State University, Thibodaux, LA 70310.

The authors have developed and implemented a microcode simulator for use in a computer organization/architecture course. This software system could be utilized in other classes to introduce students to the concepts of computer organization and architecture. A discussion of the simulator and its successful use is presented in their presentation.

The simulator has been implemented on the Apple Macintosh and MS-DOS machines. The simulator provides the user with a hypothetical machine that illustrates the fundamentals of microprogramming while executing user written code. The microcode may be read from a file or interactively written and edited using the simulator. The simulator also provides several debugging aids in an attempt to make writing microcode as painless as possible. The command/instruction interface should be readily understood by those who have programmed in BASIC.

Based on the successful use and popular acceptance by students of the simulator, the authors feel that it would be useful to others teaching topics related to computer organization at both the college and precollege level. Those wishing to obtain a copy of the simulator (Apple or MS-DOS) with documentation and source code may contact Ted Mims at the address at given above.

NEW DIMENSIONS FOR DATA EXPLORATION IN SOCIAL STUDIES

Bryant Patten
Learningways, Inc.
222 Third Street, Suite 2300
Cambridge, MA 02142

Ian Jukes
Department of Education
Yukon Territorial Government

ABSTRACT

Apple Computer's Knowledge Navigator concept is moving closer to reality. Two evolving paths of information technology, electronic databases and tools for exploring vast amounts of data, are about to merge and allow unprecedented access for the students and teachers.

Unlike any other time in history, we have instant access to incomprehensible amounts of data. Currently approximately 30 billion new documents are produced each year in the US alone and by 1990 it is anticipated that the amount of new data available will be doubling every 20 weeks. Databases of this information exist in a variety of formats including remotely accessed online systems, CD ROM devices, or even thousands of relatively small databases on floppy based systems. Teachers and students have access to financial, political, and demographic data which they will use to analyze real world events.

Combined with such a database, a multi-media authoring system for educators and students can create courseware which allows students to study a complex issue (The Civil War, Immigration, The Nuclear Arsenal, etc.) from a variety of perspectives simultaneously. Students are encouraged to move through the issue using a variety of intuitive tools and to build their own perspective on a topic by creating their own views into the data, generating their own questions as well as responding to the questions of others. Integrated presentation systems then allow students to describe their findings to the class. Such an authoring system will be presented and the implications for such an authoring systems extending computer use in the social sciences will be discussed.

We will discuss in this session both aspects of this issue and how they will effect teaching the classroom. The evolving role of teachers will be examined as they become more coaches for students becoming their own knowledge navigators.

USING VIDEODISCS AT VARIOUS LEVELS

Dean Zollman
Kansas State University
Department of Physics
Manhattan, KS

ABSTRACT

A videodisc offers interactive possibilities which cannot be achieved with other forms of video such as videotape of broadcast television. Each videodisc contains up to 54,000 individual pictures (frames). Because one has random access to all information on the videodisc, a teacher or student can view any one frame with a few simple keystrokes. At most, the search time to a frame will be a few seconds. For motion sequences students may use viewing speeds which range from very slow to faster than normal. Thus, the videodisc provides educators with many options for interactivity with visual media.

Since the videodisc was first introduced approximately ten years ago, educators have explored many ways to use this interactive medium for teaching. The types of videodiscs for education include laboratory experiments, analysis of "real-life" situations, problem solving techniques, and the storage of large quantities of individual pictures (such as astronomical pictures of collections of paintings). Each of these types or videodisc has been used by students at many different levels of education.

Vast improvements in the delivery hardware systems have also taken place over the last ten years. Stand-alone videodisc palyers can be used in many situations. However, a videodisc palyer can be easily connected to a computer with a serial port. Software developed specifically for the creation of videodisc lessons allows the educator the means to create materials for student use.

In this session several exemplary projects will be presented. They will include interactive videodisc materials used at the elementary, middle, and high school levels as well as college science teaching and teacher training. Discussions of the use of Hypercard on the Macintosh, IBM Infowindow, and stand-alone players will be included. The presenters will discuss some advantages of the various systems and of the different types of instructional strategies.

PRESENTERS

Robert Sherwood
Vanderbilt University

Robert Mohl
Apple Multimedia Group

Margo Nanny
Apple Multimedia Group

PAPER LIKE INTERFACE FOR EDUCATIONAL APPLICATIONS

Doris Chow and Joonki Kim
IBM Research Center
PO Box 704
Yorktown Heights, New York 10598

Abstract

Over the years, we have been working with special computer I/O devices consisting of a flat display and electronic tablet integrated into one surface. Along with display/tablet hardware, we have also developed software to display pen positions and means to compare one writing against another. We have been using such facilities to develop more natural user interfaces for applications. Education is one area that can benefit greatly from this new technology because of its directness and handwriting interface. We are developing a number of educational applications that make use these characteristics. Two of them, learning to write, and cross-word puzzles, are discussed in this paper.

Introduction

Human-computer interfaces have greatly evolved over the years. We have progressed from punched cards to typewriter-like terminals to high resolution graphic workstations. With each new interface technology, computers are brought closer to the general public. Connections between user and computer are becoming more intimate as we use them as our personal accountants to keep our tax records and as address/phone books. In education, they have served as able assistants from reviewing daily lessons to simulating real worlds. But still, there are areas where computers could be used more profitably if better interfaces are developed.

Dynabook concept

During the early 1970s, Alan Kay and Adele Goldberg proposed an innovative computer design which would allow people to retrieve information dynamically [1,2]. This computer system concept, called Dynabook, was envisioned as a portable, notepad-sized computer with a high resolution graphic output device. Over the display would be a touch sensitive input screen for use with a pen-like pointing device and it would have audio capability for output. Their vision was partly realized using a 64K microcomputer with a standard keyboard and a pointing device called a "mouse."

Today, their initial ideas are carried over to highly successful graphic workstations with mouse interfaces. Combining windowing techniques, pop-up menus and direct manipulation using a free-moving pointing device, it allows dynamic access of information and created a new step in user interface technology.

Handwriting recognition

In a parallel effort, researchers were working on handwriting recognition techniques since the late 1960s [3,4,5,6]. Initially viewed as a topic in artificial intelligence, recognition algorithms evolved from running on large mainframe computers in batch mode to running in real time on personal computers. Besides performance improvement, accuracy of recognition increased and different styles of writing can be interpreted. In addition to handwriting, another area of research is the recognition of handmarkings (i.e. gestures) which is used for editing and markup [7].

Electronic tablet

To gather handwriting samples and develop recognition algorithms, digitizing electronic tablets were used. Early tablets were opaque stand-alone devices attached to a computer. For data collection and evaluation of different algorithms, separation of input and output devices was reasonable. But with progress in tablet and display technology, it became possible to combine input and output surfaces into a single package [8].

Paper Like Interface (PLI)

Combining the Dynabook concept and handwriting recognition research, we have been working with a new interface called PLI (Paper Like Interface) which literally resembles paper and pen. The hardware consists of a transparent tablet mounted on top of a flat LCD (Liquid Crystal Display) and a special pen. Both input and output devices are integrated to a single surface. As the pen glides on the top surface of the display/tablet package, electronic ink is produced and a pen trace appears on the display. The computer simulates the look of paper and pen. Instead of keystrokes, handwriting marks and pen touches are interpreted as input.

Since typing is a skill which requires dexterity and special training, a writing interface offers a familiar way to interact with a computer. Writing is an over-learned skill which is acquired early and is well-practiced. With this style of interface, a user is familiar with the method of interaction and does not require learning about new input techniques.

Besides its handwriting interface, another advantage of PLI is a user's ability to directly access any area on the display surface with the pen. Since the display and input areas are physically packaged together, there is no separtion of eye and hand planes. Users point directly at a displayed object and "touch" what they want instead of moving a navigating device.

We have been using this setup to develop new user interfaces for various computer applications. Because of its natural interface and direct manipulation characteristics, PLI could help children learn letter formation and and word skills by using a computer which simulates paper and pen. Toward this goal, we are developing two packages, Learning to Write and Crossword Puzzles. Before describing them in detail, we will discuss the basic computer configuration.

Building Blocks

Hardware

Our hardware consists of a small computer, connected to a flat LCD display and a transparent tablet mounted together. The function of the tablet is to pick up the coordinates of the special writing instrument (pen) at regular time intervals. Our pen is connected to the computer.

Physical dimensions of our display/tablet package are about 11-1/2H X 15-1/2 W X 1 D, and the active area is about 5-1/2H X 8-1/2 W. Display resolution is 640 X 480. Two important characteristics of a tablet are spatial and temporal resolutions. We use our tablet at 200 points per inch spatial resolution and 84 samples per second temporal resolution [9].

Single surface input and output

Unlike a CRT-keyboard-mouse based computer where input and output devices are on separate surfaces, we have only one surface that is used for both input and output. Our input device is a writing stylus which is attached by a cable to the display/tablet hardware. To "write" on the display, a user presses down on the display with the pen and writes. Unlike the mouse, it is possible to go directly to any location regardless of the previous location without dragging a cursor. Also, it does not require extra space on a desk or table for a keyboard or mouse pad. The writing stylus feels and looks like a pen, thereby not requiring special keyboarding skills.

Electronic ink

Tightly coupled with display/tablet hardware, our software displays the pen position as the pen is touching and gliding on the tablet surface. When the pen is touching the tablet surface, pen coordinates are sampled at regular intervals. When these points are connected, we get a trace of the pen -- electronic ink. The sequence of points from pen down to pen up is called a stroke.

Stroke matching

When a stroke is written, our software compares it to a pre-stored set of strokes which serves as a good handwriting model. Models or prototypes are saved for each letter, and they can be created and changed as necessary, This comparison of an unknown to known prototypes is called stroke matching, and produces distances (or scores) as output. The name of the prototype that is closest to the unknown is the recognition result.

Matching software was originally developed for handwriting recognition. There are many ways of matching the strokes, each with different benefits. One example is a point by point comparison of normalized coordinates. Details of the matchers can be found in referenced papers [3,4,5,6].

Gestural interface

Another feature of PLI is the use of handwritten marks to indicate user actions. We refer to this type of writing as gestures or a gestural interface [10,11]. A gesture is a writing which denotes an action or object of an action. For example, a big "X" covering up a block of text could mean to delete that block of text. Similar to a macro symbol, gestural interface is a powerful and efficient way to perform commands using a natural and pictorial syntax. Gesture recognition requires a slightly different strategy [7] from our basic matching software.

Educational Applications

In our work, we successfully applied the paper like interface to a spreadsheet application [10]. The new interface proved to be more natural, direct and productive. Besides general business applications, PLI could have beneficial results in the classroom, particularly for elementary school and pre-school children.

Firstly, writing instruments and techniques are familiar to children at an early age. Scribbling and drawing are favorite activities which begin as early as toddler and nursery school ages. Through their play, they learn that crayons, pens and paint brushes produce marks and shapes. This early writing or "pre-writing" introduces them to using writing instruments.

Because it uses a writing stylus as its input device, PLI more closely matches a young child's skills than a keyboard. This style of interface can take advantage of the natural and playful way that young children interact with the world.

Currently, keyboarding and general computer skills are not introduced until students can recognize letters. This is determined by the characteristics of the primary input device, the keyboard which is letter-based. Though there is software which uses other keys or devices (e.g. space bar, cursor keys, special touch pads or mouse), these packages are limited. Most applications depend on keyboards, which rely on letter shape knowledge and typing skills.

Secondly, the bulkiness of the display and keyboard hardware can be obstructive in the classroom. A special corner and desk must be found to accommodate the various pieces of the computer hardware. Since the display and input area are a single unit, PLI is more compact and lays flat or slightly inclined. Though our current hardware configuration requires a computer attached to the display/writing unit, it is a floor standing model which can fit under a desk. Also we envision a protable unit which would have both the computer and display/writing component in a single package.

Thirdly, PLI could enable the use of the computer in learning skills which are based on handwriting. It would be difficult to use a mouse or cursor keys to perfect fine motor skills which are used in forming letters. Though using a finger to trace letters on a touch screen would help guide hand movements, the use of a writing stylus is more realistic. A student learns how to control and use a pen in forming shapes and letters.

Besides simulating the pen and pad environment, PLI offers the interactiveness and audio-visual properties of a computer. For writing-based applications, the process of writing and recognition creates a feedback loop for quickly converging on correct responses. We feel that our new interface opens a door to using the computer in early childhood education.

Two specific examples of how this interface could be applied to learning is learning to write and crossword puzzles.

Learning to Write

Learning to write is a tool for teaching children their first step in writing—letter formation. Though we are not advocates of perfect penmanship and depend on educational professionals to find its appropriate place in the curriculum, we know that children must learn the general shape of letters and perfect their fine motor skills to accomplish the task of writing the alphabet legibly.

Traditionally, teaching to write involved copying well-prepared writing materials (from a chalkboard, textbook, or worksheet) onto students' worksheets. The worksheet might have writing lines and/or dotted writing. Student writing is viewed and evaluated later by a teacher. Depending on the evaluation, the process repeats over and over again until satisfactory writing is consistently produced.

This is a time consuming process. Some shortcomings of the traditional approach are:

- Stroke order and direction are not visible in static image.
- Evaluation of student writing is a time consuming process.
- Customization of teaching material for individuals is difficult.

Our goal is to use the paper like interface technology such that teaching and learning becomes less painful.

Writing material, in the style to be taught, is presented on the LCD screen. Students trace over or copy the display. The stroke matching software that we developed for handwriting recognition can be used productively to compare the student's writing against the desired writing.

As the strokes/characters are written, the input is compared against the prototype, and the program can measure the closeness of writing to the displayed material. If the writing is out of range or exceeds some bound, the system gives audio and visual feedback to inform the student of his/her writing quality. A dialog between computer and student is established at that point and engage the student to continue or try again.

Range differences could be saved for each character, and the system could "remember" how the student is doing. This information could be saved for each student so that individual progress could be measured and managed by the teacher. Thus, we have a handwriting presentation and evaluation system, where students can practice independently and privately.

Given an interactive and individualized environment, a learning to write scenario could be as follows:

1. Prepare writing material based on student's level of past accomplishment and speed.
2. Present writing material on the LCD screen in dotted line.
3. Start the ink for the student to follow with a pen (follow-me).
4. Student uses the pen to follow the electronic ink. Display of student pen trace is different from follow-me ink.
5. As writing progresses, either at stroke or character level, compare the student writing with pre-stored prototypes and produce evaluation.
6. Depending on the evaluation, prepare new writing material and repeat with the second step until the session is over.

Variations

We have looked into one learning to write scenario. Depending on the level of instruction and student progress, there can be many variations. Some of the variations and possible applications are briefly described.

Template presentation and writing: Student writing does not have to be over the presented template. For many situations, it is better to have a writing area separate from template presentation.

Writing style and size: Our system can be set up for different writing style and size. Educators and application writers are free to choose the proper style and size to fit the instructional need. One requirement is that there is a pen lift at the end of each letter. Theoretically, it is possible to handle connected cursive writing, but we consider recognition of cursive writing still primitive and time consuming.

Composition task: As students make progress, they will be composing new materials. Rather than having the teacher evaluate student writing (in terms of character formation), the computer can recognize student writing and present the result (alphanumeric and score) to the student. If there is enough computing power, it is possible to use dictionaries and grammar checkers, too.

Adults: The scenario we suggested is overly simple for children, but for adults, it is possible to use more complicated verbal instructions.

Foreign language: The learning system we are proposing can be used for other languages as well. Because of the adaptiveness and trainability of the recognizer, prototypes are not restricted to the English alphabet. Different alphabets and notation systems can be recognized. Languages which are comprised of numerous strokes per character (e.g. Chinese), can be accommodated by our system. In fact, we can go a step further to teach notation systems such as music and shorthand.

Crossword puzzle

Another application area we have explored for children is written puzzles. One example of this class of puzzle is the crossword puzzle which is popular among adults and children. In schools, it can be used to reinforce specific vocabulary which may relate to a lesson in reading, social studies or science.

Currently, these puzzles are created by textbook and workbook companies and reproduced for class use. The task of creating crossword puzzles is difficult and time-consuming. For a high quality puzzle such as found in the New York Times, it could take a week to create a single crossword puzzle. There are even contests for finding puzzles of a specified design. Though children's puzzles are fairly simple, fitting words to a grid structure is a non-trivial task which involves mixing and matching words iteratively to get a proper fit.

This type of combinatorial processing is easy for a computer. Given a set of words, a computer can generate many different puzzles. Students can be offered a variety of puzzles for the same material, and teachers can use the automatic puzzle generation for immediate use.

Besides ease of generation, computer-based crossword puzzles can create an interactive dialog for a goal-driven task. As a game, it has a defined set of answers. Solutions are close-ended and students move toward completing the puzzle. Words and their meaning are implicitly learned in order to solve it. Being an interactive system as opposed to a static one, responses can be checked anywhere along the way and feedback is given to guide and encourage.

Entering answers for a puzzle on a CRT-keyboard-mouse based computer involves a two-step process. First, the student would need to navigate to a place in the puzzle by either pushing cursor keys from the keyboard or moving a mouse to set the cursor at a desired location. Second, text would be entered from the keyboard, and depending on how the application works, it may be a single word or character.

This process is made much simpler when placed in its natural setting such as the paper like interface. Answers are handwritten directly into the appropriate grids. Handwritten answers are recognized, and recognition results are checked against the correct answers upon request. The recognition component of a PLI application provides a feedback loop for students to get the correct answer and encourage consistent letter formation. Audio and visual feedback can be given to comfortably guide the student along.

Benefits

We have used the dynamic presentation capability to show direction and order of strokes. Using the matching software, evaluation of writing becomes automatic, which can be used to generate the next level of writing material. Each child can learn at his/her own pace, thereby easing the intense demand put on teachers in a classroom full of children. Removal of the CRT-keyboard-mouse bulk is also an important factor in classroom settings.

For crossword puzzles, we have tried to tap the flexibility of the computer more. In addition to understanding handwriting, computer is used in generating the puzzle and checking the answer. Using the computer's multiple modes of communications, clues can be displayed not only as written text, but as pictures and sounds. A computer can make the puzzle and subject matter come alive by stimulating multiple senses. Visual and audio clues are particularly helpful for beginning and non-readers and adds another dimension to the system.

There are many second-level improvements that can be added to make the learning process more interesting and educational. The purpose of this project is to show the basic capabilities. Second-level improvements can be added later.

Future work

Learning to Write and Crossword Puzzles are only two examples of how a paper like interface can be used in the classroom. Both are exploratory demonstrations which run on our PLI system. Our goal is to stimulate interest and test feasibility in an application area. Other applications which we are exploring are a music entry and editing system and putting speech and text together.

Conclusion

In this paper we have briefly described components of a computer system which simulates paper and pen.

This paper like interface (PLI) can have many uses in productively doing work on a computer. In particular, we believe that it can open new doors for the computer in early childhood education. Since it uses a familiar model, playful drawing and writing can be carried over to learning new and formal writing skills on the computer. We described two potential applications which could be benefit by the paper like interface and at the same time be meaningful to educators and the classroom. Learning to Write teaches the important task of letter formation for young children in an independent and positive environment. Crossword Puzzle can be used to reinforce words and concepts introduced in classroom lessons in an interactive manner using its natural interface. This is only the beginning of our work in using the paper like interface. By combining the flexibility and power of the computer with a more natural interface, we hope that learning can become more interesting and rewarding experience.

Acknowledgement

Many people have been and are currently involved in creating paper like interface. To these individuals, these applications would not exist if not for their hard and diligent work. Thanks to Hamed Ellozy, Steve Levy, Chuck Tappert, Tetsu Fujisaki and Tom Chefalas for their work in handwriting recognition and making the system work. Special thanks to Henry Jeanty for his work in puzzle generation and inspiring its reality in paper like interface. Equally important are our interface experts who give us guidance in this new frontier. Many thanks to Jim Rhyne and Cathy Wolf for their contributions in this field and keeping us mindful of the user's perspective.

References

[1] A. Kay and A. Goldberg, "Personal Dynamic Media," Computer, pp.31-41, March, 1977.

[2] A. Goldberg, "Educational Uses of a Dynabook," Computers and Education, Vol.13, No.4, pp.247-266, 1979.

[3] T. Sakai, K. Odaka, and T. Toiba, "Several Approaches to Development of On-Line Handwritten Character Input Equipment," Proc. 7th Int. Conf on Pattern Recognition, pp.1052-1057, July, 1984.

[4] H. Murase, T. Wakahara, and M. Umeda, "Online Recognition Algorithm for Hand-Sketched Flowchart by Candidate Lattice Method," Systems, Computers, Controls, Vol 14, No. 3, pp. 37- 46, 1983.

[5] C. C. Tappert, "Adaptive On-Line Handwriting Recognition," Proc. 7th Int. Conf on Pattern Recognition, pp. 1004-1007, July, 1984.

[6] H. Ellozy, Private Communications, September, 1986.

[7] J. Kim, "On-Line Gesture Recognition by Feature Analysis," Proceedings of Vision Interface '88, pp 51-55, June, 1988.

[8] C. C. Tappert, A. S. Fox, J. Kim, S. E. Levy, and L. L. Zimmerman, "Handwriting Recognition on Transparent Tablet over Flat Display," SID Int Symp Digest, pp. 308-312, May, 1986.

[9] J. Kim, and C. C. Tappert, "Handwriting Recognition Accuracy versus Tablet Resolution and Sampling Rate," Proc. 7th Int. Conf on Pattern Recognition, pp. 917-918, July, 1984.

[10] J. R. Rhyne, and C. G. Wolf, "Gestural Interfaces for Information Processing Applications," RC 12179, IBM Research, September, 1986.

[11] C. G. Wolf, "Can People Use Gesture Commands?" ACM SIGCHI Bulletin, October, 1986. Also RC 11867, IBM Research.

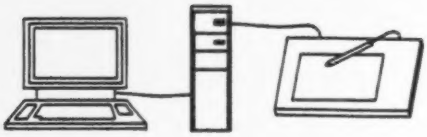


Figure 1. Hardware

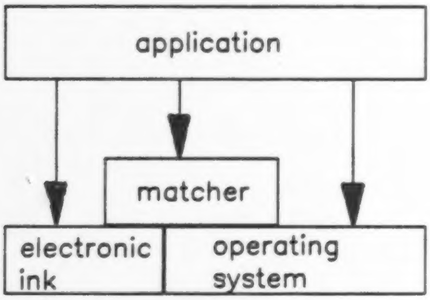

application
matcher
electronic ink
operating system

Figure 2. Software

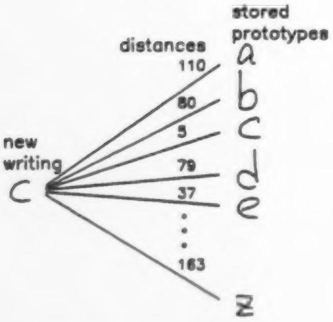
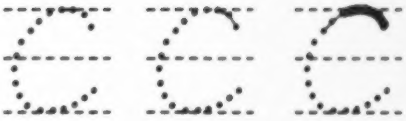

stored prototypes
distances
110
80
5
79
37
163
new writing
c
a
b
c
d
e
z

Figure 3. Stroke matching



(a) Original (b) Follow-me (c) Trace

Figure 4. Learning to write

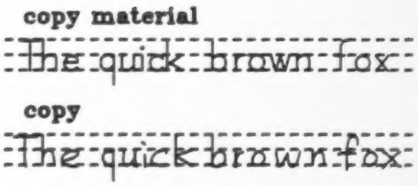

copy material
The quick brown fox
copy
The quick brown fox

Figure 5. Text copy below template

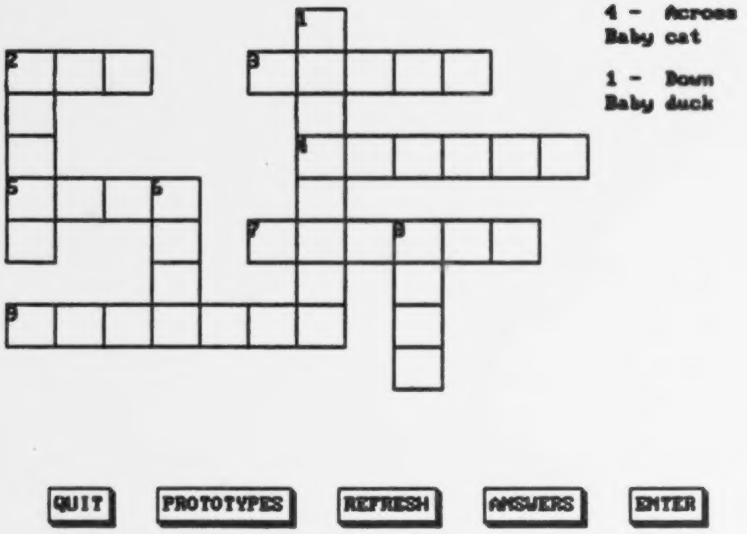

4 - Across
Baby cat
1 - Down
Baby duck
1
2
3
4
5
6
7
8
9
QUIT
PROTOTYPES
REFRESH
ANSWERS
ENTER

Figure 6. Crossword Puzzle: grid and clues

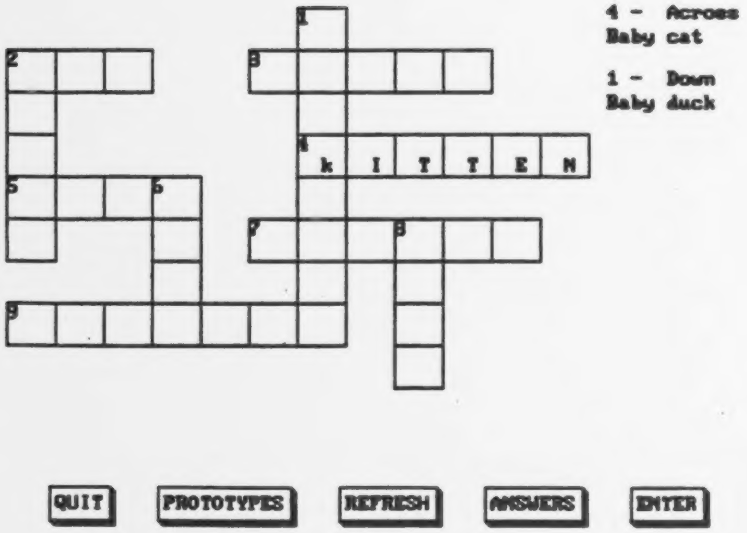
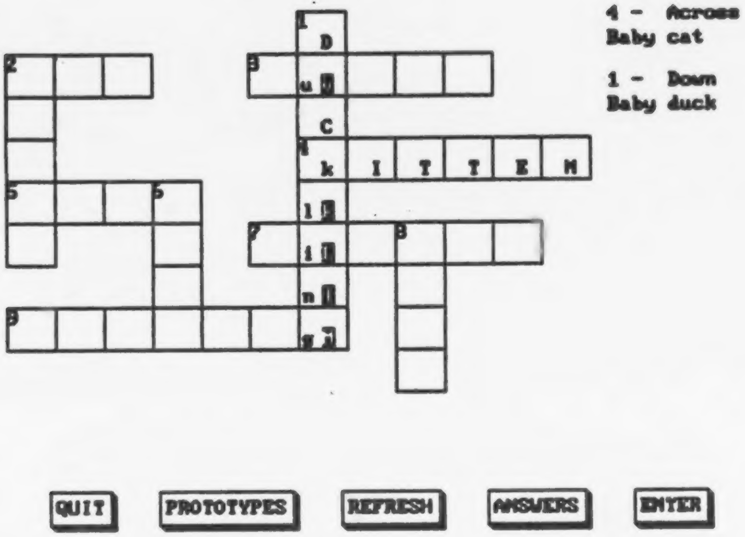

4 - Across
Baby cat
1 - Down
Baby duck
1
2
3
4
k I T T E N
5
6
7
8
9
QUIT
PROTOTYPES
REFRESH
ANSWERS
ENTER

Figure 7. Crossword Puzzle: entering and recognizing writing


4 - Across
Baby cat

1 - Down
Baby duck

D
u
c
k I T T E N
l
i
n
g
QUIT
PROTOTYPES
REFRESH
ANSWERS
ENTER

Figure 8. Crossword Puzzle - matching to solutions

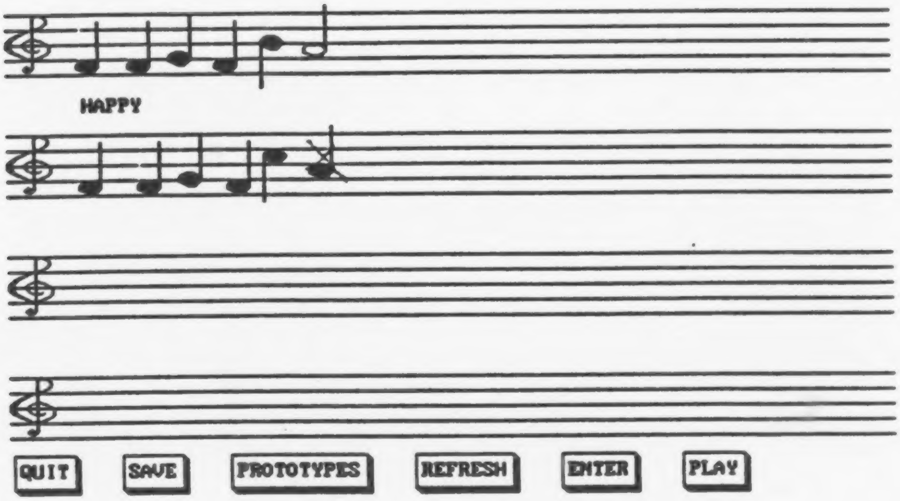

HAPPY
QUIT
SAVE
PROTOTYPES
REFRESH
ENTER
PLAY

Figure 9. Music Entry with Play Feature

INTERACTIVE VIDEO IN EDUCATION:
THE DATABASE APPROACH—A U.K. PERSPECTIVE

Wendy Hall
Department of Electronics and Comptuer Science
University of Southampton
Southampton SO9 5NH
U.K.
Tel: (0)703 592388
e-mail: wh@ecs.soton.ac.uk

Abstract

Much attention has been paid in recent years to the use of interactive video in education and training. The potential of such systems is well-recognised but very often the model suggested for their use is the 'industry training' model, which is far too expensive a solution for the majority of educational users and authors. This paper outlines the database approach to the production of interactive video material (for both the videodiscs and the software) and proposes that this model provides one solution to the problems of the cost of production of multimedia interactive teaching and learning materials for us in education. Examples from current projects in the U.K. that are using this model are presented and discussed.

Introduction

The technology of interactive video (that is the interactive control of videodisc or tape through a computer) has been available for some years now. Educators and industry trainers started by developing programs to control rather slow and cumbersome videotape recorders and have progressed to using highly sophisticated development environments and authoring languages to interactively control laser videodisc players, which now permit very fast and accurate access to any one of up to 55,000 frames on each side of a the videodisc. However, the costs of producing a videodisc with original film material and accompanying dedicated courseware are still very considerable and it requires a team of experts, including film producer, camera crew, courseware designers and computer specialists, to produce a new videodisc to professional standards. as a result, the drive for videodisc production originally came from the industry training sector. Such resources are not generally available to the education sector and other ways of harnessing the undoubted power of the moving image to enhance the ways that we teach and learn must be developed.

Another serious factor to be taken into account is that in education we are seeking to encourage the learner to develop his/her own powers of problem-solving and information sifting to achieve a desired goal. Additionally, every member of teaching staff has their own teaching style and preferred method of teaching their subject. The industry training solution does not apply in these situations, particularly in the U.K., where at all levels of education there is far less emphasis on teaching from set books and members of staff prepare their own teaching notes and paths through the syllabus. In order to make effective use of interactive video technology in education we must develop more flexible software techniques and create different models for the application of video material in different teaching and learning situations.

This paper describes the database approach to the use of interactive video in education and proposes that this approach provides one solution to the problems outlined above. A number of projects based on this model are currently underway in the U.k. and the software design techniques that are being explored to allow easy preparation and modification of curriculum material that incorporates video information are presented and discussed.

Background

The emergence of interactive video systems is one of the most exciting developments in educational technology today. The convergence of computing and video technology promises a new and flexible tool capable of providing the learner with a rich variety of facilities to support a wide range of teaching and learning needs. The interest generated by this new field is based largely on its combination of scope and power: the laser videodisc holds up to 55,000 frames per side (or 30 minutes of linear playing time) each of which can be accessed almost instantaneously giving random access to still or moving picture sequences. Two audio tracks provide the option of sound, and the availability software control through which to construct many different modes of access makes the possibilities seemingly limitless. For a survey of recent interactive video projects and trends of use, particularly in the U.K., the reader is referred to Laurillard [1], and to Hart [2], for a discussion of current projects in higher education in the U.K.

The major drawback is of course the cost of both the hardware and the software. Consequently, most of the material currently on the market is aimed at the industrial/commercial training sector. In general, the costs of producing a videodisc designed for one particular training task together with the necessary interactive software are prohibitive for education (Hall [3]). This does not mean that there is no future for interactive video technology in education but it does mean that the strategies for development must be different to those adopted for industrial training purposes. In order to make the material more flexible and of use to different groups of students, videodiscs can be designed in such a way that they reflect many different aspects of the same subject. The term often used for such discs is *generic* , implying that they consist of a collection of video images, usually a mixture of still-frames and video sequences, all pertaining to the same general subject matter but not specific to any one aspect of that subject. In other words, generic videodiscs can be described as video databases.

There are a number of examples of 'generic' videodiscs designed for use in education already in existence. Probably the most widely known in the U.K. are the BBC's Domesday discs, which represent the culmination of a pioneering project to produce a modern day interactive version of William the Conqueror's Domesday book. These discs, together with the Philips VP415 videodisc player designed as part of the same project, have done much to raise awareness of the possibilities of this new technology. An off-the-shelf system comprising the Philips videodisc player, a BBC Master microcomputer and the two Domesday videodiscs is available nationally to both education and industry at a very reasonable price. However, the price to schools and colleges of education is heavily discounted and this has encouraged these sectors of education to become users of interactive video much earlier than they might otherwise have done. One of the Domesday discs is a database of pictorial information and census data that collectively presents a record of life in the U.K. in 1986 (900 years after the original Domesday book). The other contains information that was collected and produced by school children throughout the country, as a more personal record of community life in the 1980's.

The Domesday videodiscs are currently in use in many schools and colleges throughout the U.K. either in library or classroom based situations. They have prompted the development of many imaginative and novel projects that demonstrate the exciting ways in which such multimedia information systems can be used in education. Subsequent releases of other videodiscs for the Domesday system, for example Ecodisc (a simulation-based videodisc) and Volcanoes, have made this system even more attractive to schools although the costs of these discs are still very high in terms of school budgets.

The majority of generic videodiscs developed for the higher education market seem to be in the areas of medicine and the biological sciences. These subjects are natural candidates to fully exploit the visual power of video databases. Using videodiscs it is possible to allow students to explore material and situations that it would be impossible or too expensive to set up in a laboratory environment. The students can also use the videodiscs at their own pace and as often as they wish. There are obvious implications here for the area of medical training. The successful use of expert systems in this field can only be enhanced when video images are included in the knowledge base.

Designing the Software

The move to generic videodiscs only tackles part of the problem. The challenge of designing software to make such discs useful is still considerable. Estimates of the amount of programming time required to produce one hour of student tutorial time (of the traditional computer-based training type) vary with anything from 20 hours to 100 hours depending on the complexity of the branching mechanism and the programming or authoring language used. We have learnt this to be true in a project we have undertaken in collaboration with the Human Morphology Department at Southampton based in the University of London's "The Knee" videodisc. This videodisc is a collection of still and moving images that provide a pictorial information resource covering all aspects of the study of the knee (including osteology, locomotion, biomechanics and arthroscopy). In this project we designed learning material for anatomy students using the authoring language Microtext. The project has been successful as a pilot project but the time and effort involved in producing the programs is just too much for even the most dedicated of teachers to consider worthwhile as part of their regular teaching activities. Further projects in this area are planned but alternative access strategies must be adopted to allow both lecturers and students access to the video material to suit their own teaching and learning needs.

The menu and database software provided with the BBC Domesday discs is one example of the soft of software that must be developed. The user moves around the information on the discs by means of menus and keyword searches of the video and text databases. There are no pre-determined learning paths but the menus are designed specifically for the format of the Domesday videodiscs and cannot be used in any other way. We need to develop access software that is independent of the content of the databases, in rather the same way as expert system shells are independent of the knowledge base.

The principle of combining text, graphics and video information into a single database and providing the access software to allow cross-reference to be made between the different types of information is of major significance to the effective development of interactive video in education. The greater processing power of today's microcomputers, and the massive storage capabilities of digital optical disc technology including CD-ROM, WORM (Write Once Read Many optical discs), CD-I and DVI etc., will allow us to develop such multimedia database systems for use in education and other application areas at economically viable prices.

Another term used to describe a system that allows cross-references to be made between different types of media is *hypermedia*. Before explaining this term, it is necessary to explain the term *hypertext*, which was first used in the early 1960's by Theodor Nelson [4] to describe the idea of non-sequential writing. A hypertext system is one which allows authors or groups of authors to link information together, create paths through a corpus of related material, annotate existing texts and create notes that point readers to either bibliographic data or the body of the referenced text. With computer-based hypertext systems, students and researchers are not obliged to search through library stacks to look up references and articles; explicit links allow readers to move from one document to another rather like an electronic encyclopedia. An excellent review article on hypertext and hypertext systems (from the USA perspective) by Conklin [5] is mandatory reading on this subject. A recent issue of BYTE Magazine (October 1988) also contains some interesting articles on current hypertext projects.

Hypermedia is simply an extension of hypertext that incorporates other media in addition to text. Nelson first discuss this concept in his book "Dream Machines" in 1974 [6]. With a hypermedia system, authors would be able to link a corpus of material that includes text, graphics, video, sound and so forth. Whilst seeming futuristic now, developments in technology are moving so fast that, in a few years time, such systems could be commonplace in our universities, colleges and schools. They will be used for multimedia information retrieval in industrial applications. For example, in an automated factory, the high cost of system failure requires the minimisation of repair time. However, in such environments, it is not always possible to have personnel who have detailed knowledge of every piece of equipment. It is therefore necessary to provide a method of holding the vast amounts of maintenance information in depth, but with a facility for high speed retrieval. A hypermedia system could be used to link text and graphics references to related film of maintenance procedures stored on a videodisc. Such systems also have the advantage of being able to be used for training purposes and expert system techniques could be applied to aid the engineers to locate the required data. A similar application is found in the control rooms of nuclear power stations or nuclear submarines [7].

At Southampton, we are developing multimedia database systems for use both in education and in industrial applications. One project is being undertaken in collaboration with the University of Essen in West Germany and is producing German and English versions of a system based on IBM AT workstations and Philips videodisc players. The underlying database is a corpus of 'documents', where a document can consist of any type of information from text, graphics, video and sound to an application program such as for data analysis. Hypertext structures built into the user-interface allow cross-referencing of the material. Pictorial information is classified by key-word indexes (we are still a long way from getting the computer to identify particular objects in any given image, although this is the sort of system that we must work towards using image processing techniques), and the database includes a full-text retrieval system.

The initial aim of the project is to produce a system for use in higher education and a prototype version is already in use with physiology students in Essen. It is too early to report on the students reaction to the system but problems incurred by teaching staff have already been highlighted. These include the lack of suitable videodisc material and the necessity for the system to be able to 'capture' existing teaching materials, both text and diagrams. For the latter, a scanner is included in the authoring system to allow the incorporation of scanned images into the database. The system is designed to facilitate the incorporation of application programs that can utilise the information in the database. In this way it is possible, for example, to provide an authoring facility to allow lecturers to create tutorial programs based on material they have already entered into the system.

One of the biggest problems with using videodisc material is that although the computers can be networked, videodisc players cannot. Neither can the material on the videodiscs be edited in any way. If a student wants to access videodisc material, he or she must be using a workstation with a videodisc player and a copy of the required videodisc. This makes any large scale applications in education very expensive at the present time. However, writable videodisc will be available in time (Write Once Read Many videodiscs are now on the market) and more and more of the material which we currently store on videodiscs will be stored in digital format. In effect, the software being developed now is really being designed for the technology of the future, which will include the use of satellite technology to distribute and up-date the databases of all types.

Hypercard and Interactive Video

The Apple Macintosh provides an excellent environment in which to experiment with the development of new types of learning systems. Accepting the WIMP (windows-icons-mouse-pulldown-menu) interface as standard means that design considerations can concentrate on the structure of the database and the interrelationships between the different types of data. In the autumn of 1987, Apple released *Hypercard*, which is a seductively simple but very powerful information presentation and retrieval system. It has been described as an object oriented authoring system. The objects are buttons, links, fields, cards, backgrounds and stacks. The *buttons* define actions, for example a video picture might appear or music might be played when a particular button is activated by the mouse. The *fields* can contain anything: graphics, text, numerical data. The *cards* are what you see on the screen, the background can be consistent from one card to another. A set of cards and their links are known collectively as a *stack*. A button can be used to activate a *link* between two cards in the same or different stacks.

The availability of a *video stack* enables a videodisc player to be controlled very easily through Hypercard. This is achieved using *Hypertalk* (named in deference to the Smalltalk language where it has some roots), which is the authoring language of Hypercard. It is extremely easy to use, if rather less structured than might be desired. Its statements are very close to terse natural language commands.

Using Hypercard, we have developed a learning environment for biology students based on a 'Cellular Biology' videodisc produced in Gottingen, West Germany. The stack contains close to 200 cards, the biological content of which was designed by a third year biological student. Without any previous computing experience, the student was able to program in Hypertalk with some ease after only a few weeks work. The development of the video stack required some programming expertise but once this was available, the construction of the stack from the paper-based design was reasonably straightforward. Like any authoring language, Hypertalk does have a learning curve but it is quite possible for non-specialists to produce quite sophisticated Hypercard stacks at a lower point in this curve than would be possible in most other authoring languages. Additionally, the ease with which different types of information can be linked makes the development of stacks a very motivating activity.

Evaluation of the students' reactions to the availability of a large database of still and moving video pictures through a hypertext-like interface and its educational effect is currently being undertaken. However, as far as the teachers are concerned, one of the most difficult problems in the design of multimedia systems such as the cell biology application is the analysis of the video database and the selection of the pictures and films for a particular teaching need. Sophisticated indexing facilities and database tools are required to make this process easier for multimedia applications. Another problem that we are learning how to tackle is the problem of navigation. It is quite easy to get lost in a hypertext system but navigation aids must be easy for the author to set-up and for the user to follow. In this context, traditional navigation aids for sequential text, such as indexes and contents pages, are extremely useful tools.

Conclusions

The future for multimedia systems in education is very exciting but the enormous problems to be overcome must not be overlooked. It costs a great deal of money to make a videodisc, the demand for material suitable for use in higher education is increasing but what is not clear is who should be producing them. In the UK, we would ideally have some sort of national policy to ensure that all subject areas are adequately covered and to raise the money for videodisc projects to be established at designated subject-based centres. We cannot afford duplication of effort. As the technology becomes less expensive and more commonplace, perhaps videodiscs and CD-ROMs will be published in the same way as books are today. Many publishing houses are currently looking into electronic publishing very seriously and probably electronic journals will be the first market to expand. How much more interesting and motivating would it be to learn your chosen subject from a *video* book? The technology to enable us to build the sort of learning environments described here will be developed in the next few years. Understanding how to use them and creating the software to exploit their full potential will take us a lot longer.

References

[1] Laurillard, D. (ed) *Interactive Multimedia: Working Methods and Practical Applications*. Ellis Horwoood (1987).

[2] Hart, A. *Surrogate Books*. Times Higher Education Supplement, March 18th 1988.

[3] Hall, W. *The Use of Interactive Video in Undergraduate Teaching*. Aspects of Educational Technology XXI, Mathias et al (eds), pp 158-163 (1988).

[4] Nelson T.H. *Getting It Out of Our System*. Information Retrieval: A Critical Review, G.Scheter (ed.), Thompson Books, Washington D.C. (1967).

[5] Conklin, J. *Hypertext: An Introduction and Survey*. IEEE Computer, 17-41, September 1987.

[6] Nelson, T.H. *Dream Machines/Computer Lib* (1974)

[7] McCracken, D.L. & Akscyn, R.M. *Experience with the ZOG Human-Computer Interface System*. Int.J.Man-Machine Studies, 21, pp 293-310 (1984).

THE ROLE OF TECHNOLOGY IN EDUCATIONAL REFORM

Don Rawitsch
Minnesota Educational Computing Consortium (MECC)
3490 Lexington Avenue North
St. Paul, MN 55126

Bob Pearlman
American Federation of Teachers
Boston, MA

ABSTRACT

Educational technology has been one of the major educational movements of the 1980s. At the same time, much recent media attention has been given to the weaknesses in the implementation of technology in schools and the lack of cohesiveness of research to date on its benefits. Unfortunately, in both the enthusiasm and the criticism, technology use has been set apart as a separate phenomenon from more general discussions of improving the educational system.

Educators with experience in technology realize that it will pass as just another fad if its benefits, realized and potential, are not linked to the broader agenda for school improvement. Grafted onto schools that are already struggling, technology will never reach its full instructional potential. On the reverse side of the coin, however, educational reform could be hastened by taking advantage of technology's potential.

The issues explored in this presentation relate to the following questions:

- What is the nature of the criticism being leveled at educational technology today? What evidence is available to support or refute this criticism?
- What goals have schools been trying to achieve with the introduction and integration of technology? Have they been succeeding?
- Do these goals match up to the proposed areas of reform of the many well-known reports that have been issued during this decade? Is there a place for technology in this contemporary reform agenda?
- How have teachers fared in the current reform movement? What incentives do teachers have to push for increased use of technology in their schools?
- What impact might the reform movement have on how technology is used in schools?

COMPUTERS AS CARRIERS OF SOCIAL CHANGE: USSR, COSTA RICA, AND THE USA

Seymour Papert
Marilyn Schaffer
Learning and Epistemology Group
Massachusetts Institute of Technology

Clotilde Fonseca de Pacheco
Programa de Informatica Educativa
San José, Costa Rica

Alexey Semenov
Project Delta
Moscow, USSR

ABSTRACT

Computer-using educators, like any others, put most of their energy into issues on the scale of the individual student or the school. But the presence of computers and the ways in which they are used touch on issues of a larger scale as well—issues such as the kind of society our children will live in; issues of social philosphy. The conception of this session stems from the belief that the day-to-day questions about how to use computers in education are actually closely related to these larger social issues. An awareness of such connections will give educators new perspectives on making choices that further their own views about what the future should be.

One way to get enough distance from day-to-day problems is to look at them in the context of other countries. This session will work as a panel discussion between educators from three very different countries. The first is Costa Rica where Oscar Arias—President of the country, architect of the Central American Peace Plan, and Nobel Peace Laureate—has personally sponsored one of the most cohesive, large scale "computers-in-schools" projects anywhere in the world. The second is the Soviet Union where the introduction of personal computation is closely linked with the deep changes associated with Gorbachev's goals of *perestroika* and *glasnost*. The third is the USA. It will obviously be impossible to survey these three countries in an hour. Our hope is to provoke further thinking about the long range and global implications of our daily work with computers.

IDEAS FOR ELEMENTARY PRESERVICE FACULTY TO USE A COMPUTER IN TEACHER TRAINING

Gary G. Bitter, Chair
Arizona State University
Tempe, Arizona 85287-0111

ABSTRACT

The focus of the session is to share ideas to involve elementary preservice faculty and teachers to use the microcomputer. Teacher training aspects will be discussed. Mathematics ideas will be outlined including the use of technology in the teaching of mathematics methods. Social studies will emphasize databases and their use in teacher training. Science education will emphasize microcomputer-based learning devices with specific examples of teacher education use. Language arts and reading will be highlighted with specific software suggestions. All areas will include software suggestions including simulation, application, tool, utility and tutorial software. The panel will emphasize the role of integrating the microcomputer into preservice teacher education.

PANELISTS

Ruthie Blankenbaker
Park Tudor School
Indianapolis, IN

Craig Blurton
California State University - San Bernardino
San Bernardino, CA

Mary Hatfield
Arizona State University
Tempe, AZ

Nancy Butler Songer
University of California - Berkeley
Berkeley, CA

Shelley B. Wepner
William Paterson College
Wayne, NJ

SPONSOR: IACE

CONNECTING IN THE COMPUTERIZED COMPOSITION CLASSROOM

W. Webster Newbold
Coordinator of Computer Based Education
Department of English
Ball State University
Muncie, Indiana 47306
(317) 285-8377

Introduction

As we approach the 1990s, we are entering the second decade of widespread computing in humanities disciplines, and specifically in the area of college level writing instruction. Ten years ago, the vast majority of English departments gave little thought to computing within the discipline; today, in the vast majority of these departments, computers are commonplace, both as tools for individuals and as instructional technology. Most English departments now have computers available for use with classes, even if the facility is shared with other users. English or related units in fortunate institutions have one or more computer rooms at their disposal, with networking between stations becoming increasingly common.

How we will use these facilities in the next decade is a question that merits serious thought and debate. Our experience in the 1980s has shown that using computers as automated tutors that deliver instructional programs can be a useful addition to the curriculum, enabling us to enhance the total learning environment of our students. But as computer scholars and others have objected, the computer as tutor has a potentially limiting effect, tending to encourage students (and perhaps curricula as well) to conform to its mode and content rather than to encourage students to create their own meaning (Coyle, Thompson, and Jarchow, 1988). Certainly, CAI should and will continue to be one of the main applications of our computing capability, but I would like to suggest that the careful and imaginative development of computers as tools rather than tutors may be the most promising course to follow. Specifically, the possibilities of connecting word and text processing power with communications should stimulate new ways of approaching both the available computer systems and our writing pedagogy.

Two Approaches to Word Processing

Two contrasting but not necessarily conflicting orientations toward computers and word processing have emerged (Sudol, 1987). The first attitude sees them as instrumental in achieving an accurate version of a written text; one types text into a computer at one end and it produces a clean printout at the other. In the instrumental view, the text processing and reforming capabilities of the computer make it an enhanced typewriter, different in degree but not in kind from its predecessor.

The second attitude regards computer based writing as a fundamentally different process from traditional writing with pen or typewriter. Here, writing is conceived mostly as composing within a fluid electronic medium; words and phrases take shape as a "middle essence" between the mind of the writer and the physical world, making possible previously unimagined expansions, contractions, and mutations of text. This view is "transformational" because of the way the nature of the digital process transforms our experience of writing and our vision of possibilities for teaching it.

Labs and Classrooms

It seems to me that the instrumental and transformational modes of thinking about computers and writing are directly reflected in the ways we designate and use our computer facilities. Our tendency to think in terms of computer "labs" betokens our conceiving of computers in instrumental, almost clinical terms: labs are where students produce disinfected textual products. A common pattern is to send students to the lab to "type" their papers from a handwritten draft prepared beforehand; to insist on careful editing and page correction; and to place emphasis on the formally correct final appearance of the computer printed draft. In short, students are told to do what is necessary with the technology to efficiently produce a paper or theme in clear standard English. This lab-type computer environment does indeed provide exactly what some teachers and students want for the efficient production of texts, which, to be sure, is an important function of computing.

But some of us may be further stimulated by the computer's transformational implications to rethink what we habitually do in light of what we *might* do with technology as the organizing and empowering feature of our courses. Not all teachers can or should rush to establish a computer classroom environment; in many cases, a lack of computer facilities precludes this approach anyway. But those who feel inclined to experiment with the environment and pedagogy of their classes might feel liberated by the transformational possibilities inherent in the concept of the computer classroom. In the following discussion, I would like to examine what this classroom might look like and what it might foster by way of cohesion among class activities leading to greater support for learning.

Computer Classrooms: Present and Future

Of the three common computing environments—freestanding microcomputers, networked microcomputers, and terminals attached to a central host—the computers and terminals that communicate with each other over some kind of network offer the best features for supporting a computer classroom. Linked micros, such as MS-DOS machines on one hand and Apple Macintoshes on the other, are popular in universities as well as in offices for their flexibility and general ease of functionality. Printing and software management are assisted by networking, as are the newer facilities for the sharing of two or more screen displays and files between separate stations.

If a local network of micros with specially designed software is not available, the older model computing environment consisting of terminals connected to a central host has possibilities when considered as a computer classroom. The central computer in this case is usually a powerful one, whether a mini such as Digital's VAX or an IBM mainframe. Sophisticated software can be run more easily on larger machines, and file storage is fast and reliable. Moreover, communications are frequently built into the system in the form of an electronic mail utility. Users can talk back and forth, and usually transfer texts, with ease.

One practical advantage often found by composition oriented users in larger centralized systems is that many word processing, printing, and even file and screen sharing options are already present and maintained by the computer operations department and systems programmers. On the other hand, networking microcomputers may be a frustrating and expensive proposition for a department of English; computers, software, and printers need to be made to work together, and the resulting system needs to be maintained. Although centralized terminal-host linkups might also have problems, such as user-unfriendly interfaces and frequent system slowdowns, they possess some real potential as a unified environment in which to build a writing classroom.

Software for Network Based Writing

Recently, we have seen the emergence of software supporting a wider range of writing activities that extend the basic power of word processing. The exchange of files and displays is becoming more feasible with the development and distribution of such programs as "LiveWriter" for the Macintosh, developed at the University of South Carolina, and "Forum" for DOS networks implemented at the University of Texas at Austin. ENFI (English Natural Form Interchange) is a screen sharing system originally developed at Gallaudet University, University of Texas, and several other institutions for use by hearing-impaired students; it reaches beyond the isolated classroom to connect it with students working at other universities. These kinds of facilities are beginning to empower teachers to organize classes around the concept of individual writers (including the teachers themselves) in constant communication with each other.

Although still in its beginning stages, another promising software advance is the development of hypertext programs, which have appeared on the academic scene concurrently with new and powerful microcomputers such as the Macintosh SE and Macintosh II, and the IBM Personal System/2 series. Hypertext is text functionally connected with other text or communication media; again, networking multiplies the usefulness of individual computers as writing tools when it enables users to access remote text or media resources such as knowledge bases or on-line library catalogs.

Research based writing benefits especially from this extension of the writer's reach beyond the classroom. Such an integrated software system known as "The Interactive Notebook" is currently undergoing development and testing at the University of Southern California. The "Notebook" uses hypertext via Apple's Hypercard software to organize information serving as the basis of writing assignments and connects the students to an on-line library database to search for related sources. This program will undoubtedly spawn others and increase substantially the opportunities for building a stimulating composition classroom environment around connections with external data and knowledge.

Teaching in the Computer Classroom

Computers or terminals networked in a classroom cannot in themselves offer any significant enhancement of writing instruction; but when joined with appropriate teaching methods, a computer environment can help both teachers and students to approach writing in fresh ways. In the English Department at Ball State, we have been attempting to integrate computers into the teaching of writing since 1984. Two computer classroom facilities are available to teachers on a first-come-first-served basis. One is a room housing 24 Apple IIe microcomputers without networking, the other a room with 24 Digital Equipment VAX terminals operating on the campus computer network. Both locations are popular, although the limitations of the Apples in size and connectivity restrict what can be done in that room. A document processor, several text editors, and electronic mail facilities are available through the VAX system, and these can be used concurrently for communicating messages and textfiles. Linguistic aids such as a spelling checker and on-line thesaurus are currently available from within the document processor; we hope that soon the VAX Grammar Checker will also be accessible. This program flags apparent sentence abnormalities such as run-ons and fragments and forces students to evaluate their language choices.

Many of the teaching activities taking place in the computer based writing classroom are similar for both micro and central computing environments. One set of options for building up both computing and composing skill can be called modelling exercises (Sudol). In these, students might enter error-ridden practice sentences, then edit and correct them for accurate grammar and syntax; they might take scrambled sentences in random order and unscramble them to discover principles of coherence. In order to experience the computer's ability to manipulate space, and to study cohesion within a passage of text, students can "explode" paragraphs into a collection of separate sentences, then reassemble them; in still other exercises, students might be given a passage containing frequent blanks which they must fill with an appropriate word. As they use the fluid text environment of the display screen to form and reform text, students confront patterns and recognize strategies; they gain more power over their language through their extensive experience in the word processing environment.

Using Networking

Networking enhances many functions of the computer classroom by providing the instructor with an excellent way of distributing, collecting, and monitoring exercises and assignments and by facilitating collaboration among students.

Communicating with classes over the campus VAX System, either through single messages or group distributions, has been very helpful to me in managing assignments coming in and going out and in keeping track of individuals between class meetings. I have answered questions and solved problems through electronic mail and I have read and commented on drafts in progress; I have adjusted assignments, scheduled activities and individual conferences, and consistently kept in touch with nearly every student in a class group. The communication aspect has been especially valuable with classes meeting only once or twice a week.

Many of us who teach (or have ourselves learned) with computers find that working with the technology encourages collaboration—between teachers and students and among students themselves. A computer or terminal network offers more opportunity for collaboration in the classroom at every level. Even seemingly mundane activities take on a freshness when attacked collaboratively. For example, style exercises which expand kernel sentences with bound and free modifiers might be useful in any context, but worked on collaboratively through computers I've seen them become creatively and even hilariously effective in responding to the lesson content (as in the following example):

> Kernel sentence: The woman broke the scale.
>
> Instantly, the woman broke the scale.
> Saddened, the woman instantly broke the scale.
> Saddened, the woman, who weighed 400 lbs., broke the scale.
> Saddened, the woman, who weighed 400 lbs., broke the scale that I just bought.
> (from a Nigerian student) In my tribe, saddened, the woman, who weighed 400 lbs., broke the scale that I just bought.

Further opportunities for collaboration might include the sharing of essay drafts among students working in the same subject area, collaborative revision of a single text, and the exchange of diary entries about writing experiences; these are a few examples of the kinds of larger assignments that would be enhanced by a writing classroom of interconnected workstations. But one of the greatest advantages of networking is its aiding in the creation of a common purpose and a more natural sense of audience than is usually found among composition students. Giving each person the ability to respond to assignments, comments, questions, and challenges delivered through the network encourages a greater participation by students and gives some a fresh start in dealing with their writing problems.

There are, of course, disadvantages to teaching with a computer classroom. Equipment breakdowns can be irksome; if these happen very often, the class's morale sinks rapidly, to say nothing of the teacher's. It is also necessary that the teacher have a high degree of familiarity with the particular computer system being used for the class; such expertise demands much time and effort which frequently must be invested over and above a teacher's normal work load. And, of course, the available computer facilities are never quite what we need. Yet, I believe that the positive factors outweigh the negatives when we consider the possible gains. The interchange of ideas and texts between student peers; evaluative commenting and countercommenting between teacher and student; the accessing of powerful linguistic aids by a student and tutor; and flexible management of records and assignments: many of these opportunities can be activated when we think of our room full of computers as a functional environment for learning.

Some Preliminary Results

Since the spring of last year, I have taught three such classes based almost entirely in our VAX terminal room. These have been standard composition sections, whose members did not particularly choose a computer based course. The first sets of students' evaluative comments, and my own experience, have convinced me that teaching writing in a computer classroom environment improves the overall learning process in several important ways: students feel more positive about writing, and I as teacher am able to spend more time and energy working WITH students rather than, as the standard classroom format often encourages, talking AT them. Students' good feelings about their writing seem to come in part from the additional control they have over their text as they enter, format, store, and print it; they also value the opportunity to control their ideas as they revise and edit using the comments of the teacher and their peers. A networked computer classroom allows these interactions to happen more smoothly; and when writing and commenting are focused in the computer medium (which can be a significant portion of class time), I or any teacher can use the opportunity to work with individuals on their essay drafts or on any phase of the writing process.

Students' written comments have been broadly favorable toward using computers intensively in writing class. When a recent class evaluated the course anonymously and were asked to comment on the best aspect of the course, half volunteered that working with computers to write their essays and exercises was what they liked the most about it. Also, about a third of the class remarked that the individual contact they had with the teacher was one of the best aspects of the course. When prompted by the question, "Has your attitude toward writing become more positive during this session?", thirteen students out of twenty in a different section responded yes or strongly yes; fifteen students in this same group responded that they would definitely like to take another computer based course.

Although the preceding evidence is impressionistic, it suggests, I believe, that the enhancement of the learning environment in a carefully conceived and implemented computer classroom is a real and important phenomenon. Students make important connections there that are more difficult to make elsewhere--between their thoughts and texts, between themselves and their peers, between their desire to communicate and their ability to do so, and between their inexperience and their teacher's expertise. Certainly, more formal research needs to be done regarding teaching and learning in computer classrooms, especially as it relates to the effects of networking. A recent article in *Computers and the Humanities* suggests perceptively what kinds of questions will need to be tackled; it makes clear that research in computing and teaching writing must itself take into account the synthetic nature of this new discipline with its connections with language study, rhetorical theory, communication theory, psychology, computer science, and brain research (Selfe and Wahlstrom, 1988). The computer classroom will long involve us in challenging and rewarding research across disciplines.

The Computer Classroom and Pedagogy

Finally, the dynamics of the computer classroom seem to raise two important pedagogical considerations, the first involving our thinking about connections between the stages of the writing process. The more that brainstorming, prewriting, drafting, criticizing, editing, and finalizing are centered in a word processing environment, the more natural and seamless the whole process becomes. Since electronic text is so fluid, the very words and phrases from early stages can be transferred by copying and built into later drafts; and these drafts themselves, which are central and discrete features of the traditional composing scheme, become subsumed into a continuously emerging, interlocking text. Likewise, revision and editing, through the basic functions of word processing, are drawn continuously into text production. Thus, revision of content becomes more connected with the techniques of producing text. I think that such an approach to word processing has implications for the ways we teach writing, and forces us to face questions such as, Should we rethink what we mean by "draft"? Should we attempt to redefine what we mean by "revision"? How can we guide composing so that our students' computer powered textual experience best extends their natural capabilities as writers?

The second consideration involves the way we connect with our students in the learning environments we create. The theory and practice of the computer classroom strongly supports the teaching model where integration of process and goals is central, in contrast to the more traditional type of class where the teacher is the isolated focus and authority. Many instructors will probably have trouble in a computer classroom, even if she or he is quite knowledgeable about computers; the technology will demand the attention from the students that the teacher is accustomed to receiving. My intent is not to criticize or elevate either model, but to point out that if a teacher is reevaluating instructional goals and methodology, she or he should seriously consider, where possible, the advantages of a workshop-style approach in a computer environment. Working with computers may be a first step in helping students become more conscious of their own responsibility for learning and less aware (and perhaps resentful) of the teacher's authority. The computer classroom gives both teacher and students a common ally in their struggle with the complexities of writing.

References

Coyle, Kathleen, Thompson, Ann D., & Jarchow, Elaine M. (1988). Computers in education: a call for exemplary software. Journal of Research on Computing in Education, *20*, 245-256.

Selfe, Cynthia L., and Wahlstrom, Billie J. (1988). Computers and writing: casting a broader net with theory and research. *Computers and the Humanities*, *22*, 57-66.

Sudol, Ronald (1987). *Textfiles*. New York: Harcourt.

THINKING WITH HYPERTEXT

Peter Havholm and Larry Stewart
The College of Wooster

Abstract

Most computer programs designed to teach operate in what can be described as the didactic mode of teaching. They aim at transferring a limited, clearly defined set of information and/or techniques. Even computer simulations are unambiguous models constructed from conclusions made into rules about possible behaviors and consequences. However, critical thinking cannot be taught in the didactic mode. The arts of discerning problems worth solving and of deciding which information and techniques are relevant cannot be reduced to specifiable rules.

Using OWL International Inc.'s *Guide* in a classroom equipped with thirty networked Macintosh SEs, we have been able to engage students in critical thinking through a new use of hypertext. In a course titled "March 2, 1667," students used *Guide* to create simulations of the theory of intertextuality as it applies to specific seventeenth-century texts. Because students created the simulations, they were forced to commit themselves to their own interpretations and to test and question the theory. We believe that this use of hypertext represents a transferable paradigm for using hypertext as a tool to evoke critical thinking in a whole range of courses in the humanities and social sciences.

Introduction

Winograd and Flores (1987) write: "The computer is ultimately a ***structured dynamic communication medium*** that is qualitatively different from earlier media such as print and telephones. Communication is not a process of transmitting information or symbols, but one of commitment and interpretation. A human society operates through the expression of requests and promises among its members. There is a systematic domain relevant to the structure of this network of commitments, a domain of 'conversation for action' that can be represented and manipulated in the computer" (p. 176).

Computer programs designed to teach cannot operate on the hypothesis that communication is a process of "commitment and interpretation." Because many college teachers try to teach by evoking commitments and interpretation rather than by transmitting information, they are often dissatisfied with "teaching" programs.

In OWL International Inc.'s hypertext program *Guide*, however, we have found a powerful tool, easily mastered by our students, which allows them to express "commitment and interpretation" in new ways. With *Guide*'s ability to create electronic links among texts (and pictures and sounds), students are able to create hypertexts. In a recent literature course, students' hypertexts modelled the operation of a theory in literary criticism. Such models represented the students' commitments and interpretations with respect to the theory, and their models of the theory's operation allowed penetrating discussion of its powers and of the ways in which theories guide intellectual inquiry.

Computer Programs and the "Didactic Modes" of Teaching

Many educational computer programs operate in what Axelrod (1975) has defined as "the didactic mode" of teaching (pp. 10-12). Though they are highly interactive, programs like Milton Petruk's ***Graphs***, Chauncey Mellor's ***Eine Tour durch Deutschland***, or Daniel Alonso's ***Hyperpath*** (all described in ***The Apple Guide to Courseware Authoring***, 1988, pp. 8-13) aim at transferring a limited, clearly defined set of information and/or techniques.

When, as Axelrod puts it, learning objectives are "generally clear and relatively easy to formulate," such as "the mastery of a definite body of information or the acquisition of specific motor-kinetic skills or specific mathematical or verbal skills (in English as well as in other languages)" (p. 10), teaching in the didactic modes is virtually required. Though it is possible to encourage the contemplation of ambiguity in a first-year algebra class, to do so would probably be silly.

Readers will remember from their own schooling that such objectives dominate in the primary and secondary grades and, in some areas of instruction, continue to dominate beyond the undergraduate level. Daniel Alonso's remarks on medical school make this clear: "'Someone recently calculated the number of facts that a new student has to absorb in the first two years of medical school, and it's something like 47,000. Professors feel they have to give students the latest medical information. So they cram as much as possible into their fifty-minute lectures...." (***Apple***, 1988, p. 12).

Perhaps the ideal setting for this mode of teaching is a small class with an imaginative, responsive and energetic teacher whose large collection of pictures, diagrams, exercises and films or videos is ready to hand. For every student question, such a teacher has an answer that explains the material in terms of the student's concerns and interests. With illustrations, analogies to everyday experience, exercises, the good didactic teacher bombards the student with attractive and powerful metaphors, connecting the materials of the course (whether it be in writing, chemistry, political science, statistics, algebra) to what is already in the student's head. The more imaginative, persistent and patient the teacher (and the more opportunity there is for the teacher to work with students as individuals), the more likely it is that every student will master the information or the technique.

When such a teacher can put all of the imaginative illustrations, all of the exercises, all of the amusing but telling answers to the predictable questions into a computer program through which students may be enticed at their own pace, it is often possible to improve upon the quality of didactic teaching possible in an ordinary classroom (almost never small and absolutely never packed with a customized visual aid system). With programs that allow the quick creation of electronic connection among text, pictures, music and other sounds, along with imaginative interfaces that allow students to ask questions, trace references, even manipulate the information they are being offered to see what it looks like held another way, a teacher with imagination and energy, a teacher keenly aware of the range of likely student responses as each section of material is introduced, can create a program that lavishes enormous creative energy upon every individual user.

Anyone who has ever worked with an imaginative computer program, whether designed to teach or to be used as a tool, knows that such programs have personality. This is because, as Winograd and Flores (1987) put it, "In writing a computer program, the programmer is responsible for characterizing the task domain as a collection of objects, properties, and operations, and for formulating the task as a structure of goals in terms of these. The program is forever limited to working within the world determined by the programmer's explicit articulation of possible objects, properties, and relations among them" (p. 97).

Programs are controlled by their programmers' assumptions. Whatever the ultimate ability of computer programs to emulate the human ability to shift paradigms when pushed by circumstance or sudden insight, none does so now. Computer programs can be magnificent for teaching the 47,000 facts medical students must know. But it is still difficult to imagine a computer program that could teach students to problematize the assumptions which underlie its own conceptual framework.

Simulations might at first seem to be exceptions to this rule because, in such programs, a metaphor takes over to a degree that the information to be learned seems to disappear. However, reflection shows that, by themselves, simulations represent a particularly clever way of conveying information.

In Carolyn Lougee's *The Would-Be Gentleman* (1987), for example, the apparent goal is to do well according to the rules of the simulation's world. The student makes economic and social decisions which score positively or negatively in prestige points and *livres*. A student who ignores or does not understand the rules of the program's world gets low scores and loses. A student who is able to infer the rules by which choices score well or poorly gets high scores and wins. But the real win comes through mastery of the simulation's rules as facts about society in seventeenth-century France.

Such rules must be concrete rather than ambiguous or the game would not be "fair." Indeed, were the results of player decisions random, there would be no game. Though *The Would-Be Gentleman* might well *lead to* a discussion of the available evidence for the validity of its rules or of the historiographic principles assumed by its creator, it could not *be* either. Computer simulations are unambiguous models constructed from conclusions made into rules about possible behaviors and their consequences. The dice are rolled in war games when the battle odds are within certain parameters *not* to add ambiguity but rather because the programmer believed that chance helps to determine the outcome of some battles. Chance enters in because of the rules as the programmer understands them.

Many teachers find simulations a powerful means of teaching information. We suspect, however, that simulations are rarely allowed to teach alone. That is, play is most often followed or accompanied by discussion of the rules which underlie the game world and which the play has helped to illustrate, to explain and to rehearse.

Whether a program offers multiple choice questions or a vividly rendered experience of a complex world, then, teachers who wish to do other than didactic teaching must step in. Once they raise the kinds of questions that place conceptual frameworks in question (the kinds of questions computer programs, permanent prisoners of their authors' conceptual frameworks, cannot raise), evocative teaching occurs and the class enters the realm of interpretation and commitment.

Critical Thinking

As it is generally understood in liberal arts colleges, the goal of teaching critical thinking is to empower students to deal with a broad range of problems. When versatility becomes the aim, specific information, even techniques, fall into the background. Once they have memorized those 47,000 facts, medical students must choose a specialty wisely. There are no rules one can master to make that task an automatic one. But the practice of making such decisions is the stuff of liberal education.

Most college teachers aim to teach the arts of discerning problems worth solving and of deciding what kind of information and which techniques will be most effective in solving them. Therefore, our teaching in upper-level courses aims at rehearsing students in critical thinking. Our assumption is that there is no finite set of "problem-solving" techniques that can usefully be learned and drilled. On the contrary, we believe that life and all of its more important and interesting parts are *not* bound by specifiable rules of the kind that yield scores in *The Would-Be Gentleman* or victories in chess.

The preceding paragraph is a demonstration of the key assumption underlying such teaching. Some readers will disagree with our claim that most interesting human problems cannot be adequately described by the kinds of rules that govern games and computer programs, and such a view is not provably wrong. Were it the aim of this paper to assert the value of our view of things, we could only advance a set of our commitments and the reasoning and evidence which persuade us to allow them to guide our actions as teachers.

We aspire to teach our students to be able to arrive at their own reasoned commitments in the full consciousness that knowledge is probably contingent and likely to remain so. Few educated persons would argue that a single view of the world is the only reasonable or possible one, though we know no interesting persons who are not eager to argue that theirs is the best among alternatives.

One could evoke critical thought by discussing programs like *The Would-Be Gentleman* or *Hyperpaths* as the products of particular understandings of their subjects. "Why do second-year medical students need to know 47,000 facts?" one might ask. Or, "Can one usefully *know* 47,000 facts?" or, "Were there any criteria by which French people evaluated their lives in 1637 other than social prestige and wealth?" or "*How* do we now know that 'a good marriage' was important in 1637 and what 'a good marriage' entailed?"

We suspect that the programs' authors would not object to our using their work in that way. However, we believe that even more powerful than having students discuss someone else's model would be having them create their own. The decisions which go into creating a simulation represent commitments; a model is a set of interpretations actualized. It is a conceptual framework objectified, ready for discussion.

March 2, 1667

OWL International Inc.'s *Guide* program and a classroom newly equipped with thirty networked Macintosh SEs at The College of Wooster led us to what we believe to be a novel use of hypertext.

Though similar to *Hypercard* in some ways, *Guide*'s attractions for us were three. Its use on a Macintosh is quickly taught; both the program and its products are economical in their use of disk and memory space; and the program is text-based.

Guide allows one to create "hot spots" or "buttons" in texts and pictures which, when activated by the cursor, bring other texts or pictures to the screen. The most obvious sort of "button" would be a footnote. In a hypertext article, one might activate a button in a quotation to pop the quotation's source to the screen. In other words, *Guide* allows one to create sets of linked texts and pictures.

Our course, titled "March 2, 1667," used hypertext as a way of simulating the theory of intertextuality, particularly the theory as it relates to the genesis or formation of texts. Briefly, as Abrams (1988) puts it, intertextuality "is used to signify the multiple ways in which any one literary text echoes, or is inseparably linked to, other texts, whether by open or covert citations and allusions, or by the assimilation of the formal and substantive features of an earlier text, or simply by participation in a common stock of literary and linguistic procedures and conventions" (p. 247).

The theory of intertextuality hypothesizes that meanings arise not from a text in isolation but from the connection of texts and that texts have their genesis in the language, sign systems and cultural codes available from previous texts. In Eagleton's (1983) explanation, "All literary texts are woven out of other literary texts, not in the conventional sense that they bear the traces of 'influence' but in the more radical sense that every word, phrase or segment is a reworking of other writings which precede or surround the individual work" (p. 138).

For example, music behind a scene in a film may signal impending danger to an audience. But different music behind the same scene could signal romance. According to the theory of intertextuality, this would result not from something inherent in the music itself but from the knowledge of other texts (in this case, films) in which certain kinds of music are linked to certain actions. The music from *Chariots of Fire* has come to be used in many contexts, but it nearly always signifies triumph. The first four notes from the theme of *Twilight Zone*, even in the minds of nine-year-olds who have never heard of Rod Serling, are associated with mystery. The music has become a signifier in the culture.

The very use of music to signal emotion in a film is intertextual. A host of film texts underlies even the most ordinary new production. A person who had never seen a film before might well react to "dangerous music" simply with confusion or curiosity. Why does the protagonist, alone in a cabin deep in the woods, not wonder about or even seem to hear the full orchestra, playing evidently just outside her door? And moments later, when Jason bursts through the window, why do the violinists keep playing rather than coming to the aid of the victim? The use of music to foreshadow and to induce emotion is a convention available within the culture, a convention whose meanings and powers arise from previous texts.

Our course worked very self-consciously with this theory. We began with an explanation of the theory and stated that the purpose of the course was for students to use *Guide* to simulate, and thus test.

For example, within the text of John Milton's *Paradise Lost* can be embedded the book of "Genesis," other texts interpreting or retelling the Christian fall, iconography of the time, and blank verse poetry. A click of the Macintosh mouse at appropriate places will bring the embedded text to the front. Thus, the intertextual nature of the work is modeled insofar as the embedded texts appropriately represent the language, conventions, and symbol systems underlying the original text. A good simulation will, to a degree, account for the meaning, tone and form of a selected text with its collection of proposed underlying texts.

Early assignments in the course were exercises which invited students both to work with the theory and to experiment with *Guide*. The first assignment asked students to select any text or piece of text and to embed what seemed to them underlying texts, to explain why they viewed the embedded texts as underlying. The assignment could be answered simply by a letter from the student newspaper which responded to an earlier letter, if the earlier letter were embedded and if the student explained the reason for embedding the original letter.

Later assignments became more focused and more sophisticated, clarifying the specific aims of the course, particularly as they related to the course title. The choice of the title, "March 2, 1667," was not an attempt at the cute or ingenious but resulted from a decision to test the theory precisely. The aim of the course was not to arrive at generalizations about the period of the English restoration or even the 1660s (although these did result) but to simulate intertextuality at a specific time with specific texts. Thus, students were given the entry of March 2, 1667, from the diary of Samuel Pepys and read the play, John Dryden's *Secret Love*, which Samuel and Elizabeth Pepys attended that evening.

We chose to center our study in a place and time so far removed from our own because the need for underlying texts is most obvious in a distant culture. The underlying texts of our own culture operate silently as we write and read contemporary texts. Difficulties in understanding seventeenth-century texts point to unfamiliar texts that underlie the text's formation. At first, our students could not make sense of Pepys' use of the word "mad" to describe the "mad girle" Nell Gwynne played (on the evening of March 2, 1667) in the part of Florimell in Dryden's *Secret Love*. The class had to marshal half a dozen earlier texts before they were confident of the precise blend of uninhibited gaiety and sane refusal to abide by conventional norms that was available to Pepys as a connotation of the word "mad" when he used the word as he did.

As students worked on hypertexts related to these texts, three main lines of activity and questioning emerged. The first concerned earlier texts. Students were forced to become acquainted with seventeenth-century culture in a deep and detailed way. To facilitate this, students divided into topic areas, one student attempting to collect information on science at the time, another becoming familiar with fashion in the middle 1660s, and still others working with dramatic conventions, attitudes toward marriage, the understanding of sovereignty, etc. The sharing of this information took place through discussion and the presentation of individual hypertexts within the class and through the creation of an electronic library, available to the whole class, in which students deposited all texts related to their individual topic areas.*

The electronic library became invaluable. For example, reading *Secret Love*, the class was confused by the bawdiness in Dryden's play. They could not decide whether the free-wheeling young woman Florimell was being presented as sympathetic or as a negative example. Could a seventeenth-century character be seriously proposing what sounded like an open marriage? Students soon discovered that others had deposited in the library texts from Francis Osborne, Sir Walter Raleigh, Margaret Fell Fox, and Aphra Behn which indicated diverse seventeenth-century attitudes toward marriage and which could be viewed as underlying the formation of the character Florimell.

The second line of questioning concerned the theory itself. The question essentially became "what counts as an underlying text?" Even so seemingly apparent an observation as that an underlying text must exist prior to the original text was not immediately obvious. Students of literature are so accustomed to the use of commentary and footnotes from secondary sources that it took some time for our class to internalize the recognition that a twentieth-century comment on the use of blank verse in the early 1600s could not underlie Milton's use of blank verse in *Paradise Lost*.

Other more subtle and difficult variations of the "what counts" question emerged. For example, did some kind of influence need to be proved? Must it be proved that an author had read the earlier text? Do authors figure in the theory at all? Specifically, for instance, could a comment by Pepys ever be considered an underlying text since no one in 1667 would have had access to the private diary, not published until 1825? Could non-written conventions be considered underlying texts? For example, several characters in Act 1 of *Secret Love* are described as wearing masks. As students discovered that in the middle 1660s women were increasingly wearing masks in public, the question became how or if that custom could be viewed as a text.

These questions called for a wide-ranging discussion of the implications of the theory and an attempt to arrive at provisional answers, answers always capable of revision as new questions arose and fresh insights were suggested. For instance, Roland Barthes' comment that "the quotations from which a text is constructed are anonymous, irrecoverable, and yet *already read*" (77) led to the conclusion that the texts being discovered might not be thought of as *the* underlying texts but as simply representative of such texts. Perhaps, then, some students concluded, Pepys descriptions of women wearing masks might be viewed as an underlying text because it represented a cultural code generally available in 1667.

The third line of questioning concerned the making of the actual simulation. An early problem was the practical one of getting texts into the computer. We gave students digitized versions of Pepys' March 2nd entry and the whole of *Secret Love* as well as selections from John Milton's *Paradise Lost* and Thomas Sprat's *History of the Royal Society*. However, without sophisticated scanners, students could hardly be expected to enter whole plays or epic poems.

---

* There was initial hesitation on the students' part when we proposed that they share their hard-won texts. We solved the problem by having everyone initial the texts he or she had contributed to our digitized "library," thus ensuring that we would know who had done what and (a surprising side-effect for the students) that everyone would know whose texts were found most useful by the class as a whole.

They soon found, though, that part of the critical activity resided in making choices, selecting appropriate passages, and arguing for the relevance of those passages. Further, a surprisingly large number of codes can be said to underlie even short passages. A substantial amount of sixteenth- and seventeenth-century science underlies Florimell's remark: "You wrong him, he's as constant as the Sun; he would see all the world round in 24 hours" (*Secret Love*, 1667, p. 126). Also, we discovered that twenty students and two instructors, when working over a semester and sharing all material, produced a substantial number of digitized texts and images.

A second problem related to the tension between the "realism" of the simulation and its user friendly nature. On the one hand was the desire for a simulation in which texts opened into other texts without the interruption of intervening commentary or instruction. On the other was the need to give readers sufficient guidance to make clear their choices and to explain what was being seen. The class soon came to realize that any relationship between texts is arguable.

A third problem in the making of the simulations concerned where to place underlying texts. It was clear enough that previous uses of the term might be placed under the word *Mamamouchi*, but it was less clear where to enbed previous poems that represented, for example, the convention of blank verse.

However, it was precisely the attempt to grapple with these questions and problems that provided the value of the class and of the computer simulation. In speaking of next steps in the use of the computer in education, Donald Harward, Vice President of Academic Affairs at The College of Wooster, has asked, "Could computing be less connected to conventional educational pedagogies which enhance what we already do, and more connected to creating uses and contexts which make clear the logics of learning and which clarify the structure of an intellectual activity?"(4). The need to interpret an abstract theory, frequently encrusted in jargon, and bring it to the level of trial and error undoubtedly forced a clarification of that intellectual activity.

The course, March 2, 1667, asked students to create their own simulations, not to use those created by others. To do so, students had to clarify their own assumptions, make reasoned judgments, argue for those judgments, revising them when necessary, and determine clear and appropriate ways of modeling those judgments. They had to understand, question, test, and adapt a theory that, so far as we know, has never been modeled before and probably could not be without the use of hypertext. In sum, students had to think critically and to understand what critical thinking meant.

In general, most students seemed to perceive the course as having been a valuable experience. On the final day of class, students were asked to complete an anonymous course evaluation, one question of which asked them to compare the course with others at the college. Of sixteen evaluations, twelve placed the course in the upper ten percent of courses taken at the college (the highest category), one placed it in the upper thirty percent, two rated it in the upper fifty percent, and one in the lower fifty percent. Some students added commentary, several saying it was the best course they had ever taken and some indicating that they would like to take other courses using hypertext.

Clearly, however, students found the course somewhat unsettling insofar as it disturbed their usual categories and expectations. For example, assignments could not be given in the traditional format; that is, students could not be asked to complete a three-page hypertext or a ten-page hypertext since the concept of pages is irrelevant to a hypertext, at least of the kind we were creating. Although the question of length in pages seems minor, a number of students commented on the difficulty of determining what was expected or necessary.

As well, the course itself could not be neatly categorized. Was it a course on the seventeenth century, on hypertext, or on intertextuality? Although students recognized the interrelationship of these three aspects of the course, the inability to place the course in a traditional subject-matter category was disconcerting to some. Ultimately, though, the majority of students commented on coming away with something of value. As one student put it, "I feel enriched because I have already seen how the intertextuality theory can aid my thinking beyond this class."

Perhaps the clearest argument for the use of hypertext simulations to encourage critical thinking was made by the student who wrote the following: "Active intertextuality occurs when the reader discovers the relationships by thinking why certain texts relate and then establishing the links. Therefore the existence of a hypertext, even the universal hypertext, is not nearly as valuable as the process of creating one."

It was our hope that the process of creating hypertexts would accomplish what this student went on to report that it did do: "It prompts people into challenging conversations about the origin of a text or idea. It pushes people to find the best way to make connections between one text and another, sometimes providing interesting insights not yet discovered by those discussing the question. It also gives people freedom to imagine how links could best be made with the present technology and how technology could improve the way in which relationships are now drawn."

The reader acquainted with 20th-century thought about structuralism will see that our procedure could be readily adapted to the study of any human artifacts and perhaps much human behavior. Anthropologists, psychologists and historians, as well as literary critics, all find themselves trying to teach students how to explain segments of human activity in terms of underlying cultural codes. The method we have sketched here allows such teaching to be particularly evocative and powerful.

References

Abrams, M. H. (1988). *A Glossary of Literary Terms.* New York: Holt, Rinehart and Winston.

The Apple Guide to Courseware Authoring. (1988). Cupertino: Apple Computer, Inc.

Axelrod, J. (1975). *The University Teacher as Artist.* San Francisco: Jossey-Bass.

Barthes, R. (1971). From Work To Text. In Josue V. Harari (Ed.), *Textual Strategies: Perspectives in Post-Structuralist Criticism* (pp. 73-81). Ithaca: Cornell University Press.

Dryden, J. (1667). Secret Love, or The Maiden Queen. In H. T. Swedenberg, Jr., E. Miner & V. A. Dearing (Eds.), *The Works of John Dryden,* 1966, *IX.* Los Angeles: University of California Press.

Eagleton, T. (1983). *Literary Theory: An Introduction.* Minneapolis: University of Minnesota Press.

Harward, D. (1988, May 24). *Executive Technical Briefing: Apple Computer, Inc.* Cleveland, OH.

Lougee, C. (1987). *The Would-Be Gentleman.* Santa Barbara: Kinko's Service Corporation.

Winograd, T. & Flores, F. (1987). *Understanding Computers and Cognition: A New Foundation for Design.* New York: Addison-Wesley.

MENTAL MODELS AND THE DESIGN OF INTELLIGENT TUTORING SYSTEMS

John Frederiksen, Barbara White
BBN Laboratories

Brian Reiser
Princeton University

ABSTRACT

Three subject-matter areas will be addressed in this session: physics, electronics, and computer programming. The papers will illustrate what intelligent tutoring systems have to offer by way of new approaches to instruction in these areas. We will describe how recent research in cognitive science and artificial intelligence has enabled the creation of a new breed of intelligent tutoring systems. These systems have the expertise to generate solutions and explanations to any problem posed by the student or teacher. Also, the reasoning illustrated by these systems is the type of qualitative causal reasoning that research reveals experts typically employ and students prefer. We will contrast the sterotyped procedural reasoning that is the focus of most traditional curricula with the form of reasoning that experts employ in carrying out "real-world" tasks—such as designing and troubleshooting electrical circuits, and creating and debugging computer programs. We will also illustrate how students' preferred mode of reasoning shares many of the characteristics of the qualitative causal reasoning used by experts. Then, two examples of intelligent tutoring systems that are based on causal models will be presented: (1) The RE-QUEST system for tutoring basic electricity and circuit theory (presented by John Frederiksen and Barbara White of BBN Laboratories); and (2) the GIL system for tutoring computer programming in LISP (presented by Brian Reiser of Princeton University).

GEOMETRIC SUPPOSER URBAN NETWORK:
A MULTI-CITY ELECTRONIC LINKING OF URBAN MATH TEACHERS

Grace Kelemanik
Education Development Center (EDC)
55 Chapel Street
Newton, MA 02160

Pat Potocny
Dobbins Area Vocation Technical School
Philadelphia, PA

Tom Radford
Rio Mesa High School
Oxnard, CA

ABSTRACT

In 1985 the Ford Foundation sponsored the establishment of eleven locally controlled projects that seek to strengthen mathematics education in urban high schools. These eleven Urban Mathematics Collaboratives (UMC) are organizations through which high school math teachers can learn about new mathematics and changes in mathematics education and in which in which they can work together with other teachers in their schools, their districts and other school systems. Through such collaboration UMC teachers have begun to break down the physical and intellectual isolation that are characteristic of teaching and to gain a renewed sense of energy for the work they do. One vehicle through which this breakdown of isolation and sense of professional empowerment has occurred is an electronic communication network for UMC teachers called the Geometric Supposer Urban Network (GSUN).

The GSUN grew out of a desire to support teachers from the Los Angeles, Memphis, Philadelphia, and Pittsburgh collaboratives who were teaching geometry with the help of *The Geometric Supposer* software program. Each member of the network is equipped with a computer and a modem that allow the teacher to log onto the network and discuss this new approach to teaching and learning geometry. Acting as initiators and critics of new ideas, the network members report their difficulties and successes, and challenge each other to find answers to very practical problems of implementation.

During the presentation two GSUN members will discuss, giving concrete examples, how the network has contributed to the enrichment of their critical inquiry about their subject, their students, and their profession. One of the members, a teacher from Los Angeles, will explain how the sharing of new ideas, lab activities, successes, frustrations, and information has been very valuable, making his work seem more self directed and his teaching more enjoyable. A Philadelphia teacher will talk about the ideas and suggestions from the network that have redirected her teaching strategies, increased her confidence, and linked her with colleagues who have similar beliefs and pedagogical interests. Lastly, the network moderator will address some of the issues involved in networking teachers through electronic communication systems, e.g. network setup, facilitation of online discussion, equipment difficulties.

TEACHING ASTRONOMY WITH MICROCOMPUTERS

Philip M. Sadler
Director of Project STAR
Harvard-Smithsonian Center for Astrophysics
Cambridge, MA 02138

ABSTRACT

Black holes, comets, colliding galaxies, and exploding stars, all hold high interest for students, yet none can be toyed with in the classroom. Astronomy is an unusual science in that experimentation is so difficult. To remedy this situation, astronomers have, for over 3,000 years, used a variety of ingenious computers to model the heavens. Astrolabes, mechanical calculators, stonehenges, medicine wheels, and orreries were all used to make the mysteries of the heavens accessible. These have all been replaced by computers. The astronomer without a computer on his or her desk is now a rarity. Recently microcomputers have begun to transform the teaching of astronomy as supercomputers have radically changed astronomical research.

Teaching astronomical facts is easy; teaching concepts is difficult. All students, at every age level, have well thought out theories about how the world works. These ideas, assembled from their own observations and reasoning often do not agree with their teachers' ideas, but almost never change as a result of conventional instruction. We have experimented with using computers to overcome these "misconceptions." Although astronomical software for the elementary and secondary classrooms is now in abundant supply, computerized data analysis, image processing, and laboratory controlled equipment are just now finding their way into school classrooms. These applications hold the best chance of success in teaching concepts.

This session will demonstrate and explore computer use for:

- Simulating cometary and planetary orbits.
- Astronomical programming projects for students.
- Gravitational catastrophes like galactic collisions.
- Using spreadsheets for making sense of astronomical observations.
- Showing the motions of stars, planets, and constellations.
- Analyzing light with a powerful Apple //e spectrophotometer.
- Recent supercomputer simulations of the formation of the Moon.
- Development of computer-controlled telescopes for classroom use.

The state-of-the-art tools of astronomical research will become commonplace in the classrooms of tomorrow. Participants will have the opportunity to make copies of simulations and databases.

STATE SUPPORT FOR INSTRUCTIONAL TECHNOLOGY

Michael Neill, Chair
Indiana University
Indianapolis, IN

ABSTRACT

A number of states are supporting local school district efforts to utilize instructional technology effectively and productively. Representatives from five states will report on projects and programs which include funding, training, consulting, planning, and dissemination of information. A variety of models for state support will be discussed.

PANELISTS

Geoffry Fletcher
Texas Education Agency
Austin, TX

Cheryl Lemke
Department of Public Instruction
Olympia, WA

Harold McDermott
Minnesota Department of Education
St. Paul, MN

Elsie Brumbach
Department of Public Instruction
Raleigh, NC

SPONSOR: ICCE

"WHO OWNS AN IDEA?"
SOFTWARE PIRACY, "FAIR USE," AND SOFTWARE DEVELOPMENT ON CAMPUS: FACULTY AS ENTREPRENEURS vs. FACULTY AS CONTRIBUTORS TO WORLD KNOWLEDGE

Steven Gilbert
EDUCOM
Princeton, NJ

ABSTRACT

An overview of issues dealing with the need for new rules to guide the exchange of results of scholarly work in the Information Age.

Given rapid advances in Communication and Information Technologies, there are today a number of serious questions being raised about the continued viability of many of the laws and norms that have traditionally governed information exchange. These laws include, for example, those relating to intellectual property rights, privacy rights, and the rights of information access. To have a major impact on the situation, however, universities and other institutions of higher education will need to begin now to develop a coherent and well-conceived position. The panel is intended to initiate a dialogue on information exchange issues as they relate to higher education.

Points for Discussion

The audience is those concerned with points ranging from software piracy to institutional rights to works of faculty members.

The impact and future of the EDUCOM CODE:

Respect for intellectual labor and creativity is vital to academic discourse and enterprise. This principle applies to works of all authors and publishers in all media. It encompasses respect for the right to acknowledgment, right to privacy, and right to determine the form, manner, and terms of publication and distribution.

Because electronic information is volatile and easily reproduced, respect for the work and personal expression of others is especially critical in computer environments. Violations of authorial integrity, including plagiarism, invasion of privacy, unauthorized access, and trade secret and copyright violations, may be grounds for sanctions against members of the academic community.

PANELISTS

Linda Garcia
Office of Technology Assessment
Washington, DC

Francis Fisher
Harvard University
Cambridge, MA

SPONSOR: EDUCOM

ADVENTURE GAMES: CONNECTING THE CURRICULUM

Cathie Sherwood
Department of Mathematics, Science and Computing
Brisbane College of Advanced Education
Mt. Gravatt Campus
Queensland, Australia

Abstract

Teachers of young children are well aware that the computer can be utilised effectively in their classroom as a medium for the development of skills such as visual discrimination etc. Are they aware however, of the enhanced learning which can be achieved through graphics-based adventure games? This paper looks at the use of adventure games, ways of integrating them into the curriculum and the effect they can have on learning. One graphics-based adventure game, *Detect-A-Pet* is considered in detail. *Detect-A-Pet* was designed specifically for early childhood. The design took into account the limited alpha-numeric skills of children in this age group. The package was then trialled in three early childhood classrooms to ascertain the effectiveness of a picture based, problem solving adventure with young children. Also considered in the trial was the change in role for teachers - from that of leader and provider of information to that of co-learner and facilitator. Teachers in these classrooms took up the challenge and created effective environments, using *Detect-A-Pet* as a catalyst, to transform the academic, social and affective domains of learning.

The introduction of the computer into the early childhood classroom continues to provoke discussion amongst educators. While few doubt that computers in classrooms are here to stay, there is still debate about their most appropriate and effective use within the teaching and learning process. Early childhood teachers already immerse their children in a stimulating and creative environment. However, many of the learning experiences they provide may be enhanced by realising them in the interactive context of computer-based activities such as adventure games.

Young children playing with computer programs are involved in first-hand experiences, although it is with representations of knowledge rather than with concrete materials. They rearrange the puzzle pieces displayed on the screen, they draw shapes and fill them with colour. They may move robots through unknown territories or count with an animated rabbit. Are these experiences appropriate? Some educators (Barnes and Hill, 1983; Bobrow, 1985) hold grave fears that by using the computer with young children they will dehumanize the socialising process so very important at this stage of development. These same educators fear that the computer will dominate other 'real' classroom materials such as paint, clay, colouring pencils, cuisenaire or Unifix blocks; the very concrete materials which enhance learning at this level.

Recent studies of computers in early childhood education indicate however, that this is not a major problem in a stimulating environment (Killian, Nelson and Byrd, 1986; Rosengren, 1985; Buckleitner and Hohmann, 1987). Many teachers of young children have utilised the computer effectively in their classrooms as a medium for the development of skills. These experiences have usually centred around highly structured drill and practice software and reinforced knowledge of letters and numbers using vividly colourful graphics screens and music. While this is appealing for some children, for many educators this seems to be an expensive way to occupy young learners with traditional tasks. The thematic-based, language-oriented classroom of the early childhood teacher requires more open-ended, child-centred software compliant with current early childhood learning theories (Shade, 1985).

The challenge then for these educators is to be able to take this technology a step further into the realms of interactive, thematic-based learning where the computer can be an interesting, appropriate and useful resource; connecting the curriculum.

How then can teachers of young children take up this challenge? One way that teachers of older children have enhanced the learning in their classrooms is through adventure games. Adventure games provide a rich, creative and imaginary world; one that is difficult to provide in any other way: a world in which the players can manipulate, make decisions, discuss and simulate. Each game usually presents a number of locations. At each location clues must be looked for, objects can be taken, characters met, and problems and puzzles solved. Most adventure games are text-based, with children determining each step through the use of a vocabulary command list. Common commands are Look, Take, Drop, Inventory, Go (North, South East, West), Examine and Use.

Adventure games, by their very nature, are readily integrated into the curriculum. They can be used as the focal point of a unit or simply be one resource within it. Teachers may design activities based on the adventure game which incorporate all subject areas. Thus, the thematic unit connects all curriculum areas so that one area is not viewed in isolation from another.

Would this type of software prove to be as successful with young children as it has been with older children? Would it enhance the learning in an early childhood classroom? Most definitely, the answer is "Yes", particularly if the software takes into account the limited reading abilities of children at this age. Specifically for this age group (but basic to all learning) is that children learn best when they are actively involved. Effective learning will not take place in a passive environment where the child merely receives the information (Papert, cited in Woodill, 1987). Within the adventure game, children must make the decisions, solve problems and puzzles and look for clues. There are many paths that can be taken through the game, each of which may lead to a gratifying conclusion. Therefore, the children will be fully involved as it is their choices which will determine the outcome of the game.

Adventure games do not provide immediate feedback about the final outcome of the adventure, or whether correct or incorrect paths have been taken, nor do they provide overt positive reinforcement unless a problem or puzzle has been solved correctly. Children will only realise the consequences of their decisions by failing or succeeding. How they then use the information in subsequent games is of paramount importance.

Children will learn if individual progress can be made. However, more recent proponents of microcomputer use in education have emphasised its capacity to stimulate active collaboration and discussion between children, thus promoting learning (Light, 1987). Adventure games, ideally, should be played by groups of three to five children. Children's talk is encouraged, and social skills such as cooperation and peer interaction developed when decision making is a shared process.

By designing and organising activities based on the adventure game the teacher can provide multiple experiences which can cover the aural, visual and kinesthetic learning styles of children. These activities should build on what the children already know, so that they will perceive some relationship to their real world knowledge and experiences. Within this environment, the children can experiment, be challenged, surprised, fantasise, play, solve problems, direct their own learning and cooperate with each other and the teacher. Motivation thus becomes intrinsic rather than extrinsic, as the children take an active role in the learning process (Malone and Lepper, 1987).

Adventure games can be the basis for a different way of thinking about teaching and how children learn. New strategies to incorporate the adventure game may have to be devised which support and complement contemporary curriculum goals and practices.

To this end the Department of Education in Queensland, (Australia) through the Software Development Unit, undertook a project to design an adventure game specifically for young children. The product is *Detect-A-Pet*. The producer of the package worked in a number of preschools and early childhood classrooms to observe the play and formal activities of these children. The software and supporting curriculum materials were then developed in conjunction with the teachers of these children.

One Example - Detect-A-Pet

Detect-A-Pet is a graphics-based adventure game designed for children from four to eight years of age. The package is based on the whole language model of teaching literacy, where the teacher utilizes opportunities to foster children's use of language through meaningful and interactive oral and written language experiences. Skills and understandings can be developed in an integrated, connected way across the curriculum. (Sherwood, 1988)

The situation used in the software (that is the disappearance and subsequent search for a pet) is one in which children could easily find themselves, though they may be unable to pursue the search because of fears for their own safety. Within the software, children can meet problems and make decisions within the safety of the computer environment.

The program requires children to use a variety of written and spoken skills, and to make choices through the experiences that unfold within the software story plot. It has been designed for use by individual children and small groups again to foster language and social interaction. Because young children's skills of reading, writing and alpha-numeric recognition are still developing, appropriate attention has been paid to screen presentation, communication via symbols and the directions and choices given in the software.

Detect-A-Pet has four major components - the software, a story book *Something Very Special*, a game board and the Teachers' Guide which contains two thematic units - Pets and Our Neighbourhood - and copy masters. The package has been designed so that each part supports and extends the learning that should take place in each of the other components. The parts are not designed for use in isolation but as supporting elements in creating and developing a familiar context in which effective natural learning can take place.

The central character around which the package has been developed is a child who has lost a pet. In the software, players take on this role and search a local neighbourhood to find the missing pet. No time limit is set for the search. However, certain conditions must be met before the pet can be found and taken home. In solving the problem, students have activities to do, objects to take and clues to obtain. If certain clues are not collected, and a correct solution deduced from them, players will exit from the adventure. However, players can use the skills and knowledge gained from their previous encounters to make other decisions leading to a solution to the problem the next time they use the program. In this way, players can relive the experience any number of times to test their theories. The game also has a Save facility so that players may store their game on disk at any point. The story solution may involve a different pet each time the game is played, as the computer randomly selects a pet from six choices - a dog, cat, rabbit, bird, mouse and duck.

The software is based around a neighbourhood which has thirteen locations including a Pet Shop, a Supermarket, and a Police Station. Upon arrival at a location, players will see on the screen a graphic of that location, and five icons - Look, Move, Take, Drop, Bag. Each time the players make a choice of action, it will involve a selection from these five alternatives. Icons, that is words with symbols, were chosen for interaction with the software so that even the youngest participant could gain the meaning of the words through association with the symbols.

The storybook, *Something Very Special*, is an important component of the package. It is used to set the scene for the software. In this way students are given a purpose when using the software. The Game Board is a concrete aid for students to follow the steps taken through the neighbourhood on screen. Players can match locations shown on the Move Map with those on the Game Board.

The Trial

Detect-A-Pet was trialled in three classrooms - a Year One (5-6 years of age), a Year Two (6-7 years of age), and a multi-age grouping (Years One, Two and Three). In the three classrooms, the Year One and the Year Two teacher opted for the thematic unit on Pets, while the multi-age teacher chose Our Neighbourhood.

What were the outcomes?

Problem Solving and Decision Making

The software provided autonomy, that is, the children could proceed without the constant assistance of an adult. They had the freedom to move through the game at their own pace. When the game became 'too hard' they were quite happy to save it, go away, do other things and come back to the adventure later. Some children in the single Year One class became frustrated after not finding their pet on a number of occasions. On the other hand most of the older children were capable of utilising information 'learnt' in previous games.

Peer tutoring/peer discussion was all-important. The children helped each other through 'tricky' sections. There were very few occasions where the teacher provided answers. The teachers became facilitators rather than purveyors of knowledge. Decisions were made by the children in consultation with their peers. On rare occasions the teachers guided the discussion. The outcome of this was that control of the learning situation was transferred from adults to the children. Some of the children developed an independent learning style which they also applied to other off-computer activities.

Written and Oral Language

The most powerful language learning situations are those which involve all children in a group or class interacting either verbally or through writing. Development of a child's language involves talking, listening, moving, making, thinking, valuing, imagining, remembering, as well as reading and writing.

The theme work generated by the adventure game provided a unique (and different) opportunity for oral language development. The children learnt to cooperate, listen, predict, offer a point of view, accept compromises and work towards a common goal.

Many activities took place away from the computer - activities which consolidated, extended and enhanced the experiences of the situation. The activities included a trip to the local pet shop, mapping the area around the school, a visit from a vet, as well as painting, writing, debating and graphing.

Word Banks were developed over the period of the thematic unit, with the children becoming familiar with 'difficult' words such as neighbourhood before 'easy' words such as park. *Detect-A-Pet* became a valuable adjunct to a much wider classroom thematic unit.

Social Development

The computer fosters cooperative work among students as the goal is not necessarily to complete but to problem solve. Students who collaborated on the adventure game often taught each other. Such computer-oriented group work provided a powerful socialising experience. Groups of children at the computer not only had positive social ramifications but it was also advantageous for time efficiency, for peer assistance and for seeing more solutions to a problem.

Those children who were not normally 'risk-takers' took advantage of the support provided by the peer network. Once the concept of the adventure game became clearer to the class, there was greater cooperation as the children realised that a decision made in one game need not be made in a subsequent game if that decision proved to be incorrect.

Interest levels were heightened and concentration spans increased particularly while children were at the computer. Teachers felt that this was not always the case with other activities such as maths games. Motivation for reading and writing activities was increased, especially for those children with learning difficulties.

A Stimulating Tool

The trial teachers provided stimulating environments for the children. Resources of all shapes, sizes and types were utilised to focus on the themes - Pets or Our Neighbourhood. Learning had a real purpose. Classrooms became a mini-neighbourhood and a mini-pet shop. There was environmental print, poems, charts, vocabulary lists, games, and stories (most of which were done by the children).

Computer use in this stimulating environment left students feeling more in control of their learning (I don't want Mrs Radcliffe to help. I can do it on my own, thank you. Brad (5) a 'slow learner'), of knowing they could play the game at their own pace, of using 'trial and error' as a strategy and seeking alternative ways to solving a problem through discussion with other members of the group. Most of the children were willing to make a decision and take risks. The teachers in the trial classes allowed this freedom in a stimulating environment. They questioned rather than informed, guided rather than led.

Conclusion

If the use of adventure games in early childhood classrooms is going to become more commonplace, how then will the role of the teacher change? What can teachers do to utilize effectively this 'golden opportunity'?

This type of computer use can enhance what has been the traditional curricula for young children. A teacher who uses an adventure game as a catalyst or focus for an integrated thematic unit has created a motivating learning tool. If the computer continues to be used in this way then it becomes a recognised and accepted part of the total environment. The role of the teacher changes from an instructor to an overall facilitator and coordinator. Teachers create an environment in which the child can be an active, rather than passive, learner, where they can choose and complete their own activities.

Karen Burg (1984) believes that the computer is an educational tool and '. . . all educational tools are value neutral. They can be used to promote divergent thinking or conformity, freedom or restriction, self-confidence or fear. Choosing how an educational tool will be used is the responsibility of teachers'. Will teachers of young children take up the challenge? Their students are already immersed in an activity-based learning environment. Understandings and skills are developed in an integrated way across the curriculum. Graphics-based adventure games can be totally synthesised within this teaching/learning environment which is small group based, thematic and developmental especially in language, both written and oral.

Teachers who use computer-based experiences, particularly adventure games, will complement and enhance other learning experiences in their classrooms, through the development of attitudes of inquiry, the encouragement of cooperation, discussion and logical planning and the motivation and encouragement of problem solving strategies.

References

Barnes, B.J. & Hill, S. (1983), "Should Young Children Work with Micrcomputers - Logo before LEGO?", *The Computing Teacher*, May, 11-14

Bobrow, B. (1985), "Do we need High Tech Toddlers?", *Canadian Journal of Research in Early Childhood Education*, 1, 1, 95-99

Buckleitner, W. & Hohmann, C. (1987), "Technological Priorities in the Education of Young Children", *Childhood Education*, 63, 5, 337-340

Burg, K. (1984), "The Microcomputer in the Kindergarten: A Magical Useful Toy", *Young Children*, 39, 3, 28-33

Killian, J. et al (1986), "Child's Play: Computers in Early Childhood Programs", *The Computing Teacher*, 14, 1, 13-16

Light, P. et al (1987), "Collaborative Interactions at the Microcomputer Keyboard", *Educational Psychology*, 7, 1, 13-21

Malone, T.W. & Lepper, M.T. (1987), "Making Learning Fun: A Taxonomy of Intrinsic Motivations for Learning". In Snow, T.E. and Farr, M.J. (eds), *Aptitude Learning and Instruction. Vol 3: Creative and Affective Process Analyses.* Lawrence Erlbaum, New Jersey

Rosengren, K. (1985), "An Observational Study of Preschool Children's Computing Activity". ERIC ED 256 288

Shade, D. (1985), "Will a Microcomputer really benefit preschool children? A theoretical examination of computer applications in early childhood education" ERIC ED 264 951

Sherwood, C. (1988), *Detect-A-Pet*, Department of Education, Queensland, Australia

Woodill, G. (1987), "Critical Issues in the Use of Microcomputers by Young Children", *International Journal of Early Childhood*, 19, 1, 50-57

TEACHING COMPUTER SCIENCE COURSES WITHIN MATHEMATICS DEPARTMENTS

Judith L. Gersting, Chair
Indiana University - Purdue University at Indianapolis
Indianapolis, IN

ABSTRACT

In 1988 a Task Force on "Teaching Computer Science Within Mathematics Departments" was established by the MAA, the ACM, and the IEEE Computer Society. The Task Force was charged with examining issues of professional standards and development for insuring quality instruction in computer science.

This panel will discuss the recommendations of the Task Force as presented in its final report of April, 1988. Topics addressed will include: minimum faculty qualifications for teaching various computer science courses; suggestions for faculty development; differences in faculty workloads between computer science and mathematics; tenure and promotion considerations that may differ between the two disciplines; pros and cons of separate departments; and institutional support needed for a computer science program.

PANELISTS

David Ballew
Western Illinois University
Macomb, IL

Ben Haytock
Allegheny College
Meadville, PA

James Stager
Millersville University of Pennsylvania
Millersville, PA

SPONSOR: IEEE-CS

COMPUTER SCIENCE METHODS

Larry E. Liden
J. Philip East
Thomas Morgan

ABSTRACT: Computer Science Materials and Teacher Preparation Project

Larry E. Liden, Superintendent, the Institute of Computer Technology, 1196 Lime Drive, Sunnyvale, CA 94087.

An adequate supply of computer science graduates is critical to continued growth of our economy. At the same time, the number of high school graduates selecting computer-related courses is dropping and often those who do enter the field find their high school preparation inadequate.

One cause for this problem is the lack of qualified teachers able and willing to teach advanced computer courses. The amount of preparation required discourages able teachers from accepting the responsibility.

To help correct this problem, the Institute of Computer Technology, a public school agency specializing in facilitating technology education has developed a two-phase approach. Using funding provided by the California legislature, we produced curriculum guides in advanced placement computer science, C Language and the use of microcomputers as a productivity tool for college. These guides were developed by teams of educators and computer scientists from various Silicon Valley industries who spent over 6,000 man hours developing and testing lesson plans, overhead transparencies, lab exercises, worksheets, and handouts.

These guides are available for the cost of duplication. Although originally intended for dissemination to schools in California, we have filled orders from throughout the United States. The guides have dramatically reduced the preparation time to give the teacher more time to develop new materials, practice new techniques, and avoid the omnipresent "burn-out."

To help teachers use the materials with confidence, the Institute developed two intensive teacher workshops. We designed a five day workshop for teachers planning to prepare students for the Computer Science "A" exam offered by the College Board and a six day workshop focused on the more advanced subsets of the traditional and more advanced "AB" level exam. Both workshops were very successful and will be expanded to two locations during the coming summer.

ABSTRACT: An Undergraduate Methods Course in Computer Science

J. Philip East, Mathematics and Computer Science, University of Northern Iowa, Cedar Falls, IA 50614.

During the Spring semester of 1989, a course called *The Teaching of Secondary Computer Science* was taught for the first time as a regular class at the University of Northern Iowa. The content of and methods of teaching secondary computing courses (programming, literacy, etc.) formed the foundation for the course. Goals, content, organization, teaching activities, and feedback and grading were discussed for each type of course. Additional discussion occurred on various general aspects of teaching such as teaching and learning beliefs and activities, text selection, and classroom management.

The project presentation will describe the environment of the course; outline the content, process, and assignments used in the course; describe student responses to the course; indicate expected modifications for the next course offering; and allow for participant questions and suggestions.

ABSTRACT: Dynamic Visual Organizers for Introducing Sorting Algorithms

Thomas Morgan, Central Virginia Magnet School for Science and Technology, 3020 Wards Ferry Road, Lynchburg, VA 24502.

Understanding computer sorting algorithms is often a very difficult task for novice programmers due to the abstract nature of the subject matter. Students are required to visualize the motion inherent in the steps of the algorithm and to comprehend how the code implements those steps. Computer science instructors need to develop methodologies designed to help students deal with the abstract nature of higher levels of semantic knowledge thus facilitating its acquisition. Utilizing the graphic capabilities of the computer, dynamic visual organizers can be developed which provide students with concrete representations of the relationships between the steps of a sorting algorithm, the inherent motion of the sorting process, and the code which implements the algorithm. This project describes the development of dynamic visual organizers for introducing sorting algorithms covered in the AP Computer Science Course. Software packages used in the development process include HyperCard and VideoWorks II.

LEGO/LOGO: EXPERIENCES IN THE CLASSROOM

Stephen Ocko, Mitchel Resnick
Massachusetts Institute of Technology
Media Laboratory
Room 301, 20 Ames Street
Cambridge, MA 02139

Linda Darisse
Salem Middle Schools
Salem, MA

Michael Iovine
Minuteman Technical High School
Lexington, MA

Geraldine Kozberg
St. Paul Public Schools
St. Paul, MN

ABSTRACT

LEGO/Logo is a type of inventor's toolkit for children. Using LEGO/Logo, children begin by building machines out of LEGO pieces, using not only the familiar LEGO bricks, but also newer pieces like gears and motors and sensors. Then, the children connect their machines to a computer and write computer programs to control the behavior of their machines. For example, a group of children might build an automated amusement park, complete with a LEGO roller coaster and Ferris wheel. Or they might build a "House of the Future" with programmable sewing machines and pop-up toasters.

By building and programming LEGO/Logo machines, children encounter mathematical and scientific concepts in a meaningful and motivating context. At the same time, LEGO/Logo makes it possible for children to learn about design and invention as fields in their own right. LEGO/Logo allows children to explore design ideas in two domains: the LEGO building environment and the Logo programming environment. Each of these environments is, by itself, rich with possibilities for learning about design. When linked together, the two environments become even richer: each environment supplements and reinforces the other.

At previous conferences, we have described the development and early testing of LEGO/Logo, and we have discussed the pedagogical goals of the project. Now that LEGO/Logo is commercially available, we are beginning to learn more about how LEGO/Logo is being used in "real" classrooms, outside of a research setting. At this special session, we plan to focus on those classroom experiences, probing how LEGO/Logo is being shaped and appropriated by different individuals and schools.

The session will include an overview from two of the developers of LEGO/Logo, followed by presentations from several teachers and administrators who have been using LEGO/Logo in their schools.

SECONDARY MATHEMATICS

Catherine M. Sarther, Sharon L. Senk
Barbara Shelly, Patricia Tinto

ABSTRACT: Development of a Precalculus Mathematics Course Based on Computers

Alfred Bork, Educational Technology Center, Information and Computer Science, University of California, Irvine, CA 92717; Fran Antenore, Stephen Franklin, University of California, Irvine.

Almost all universities in the United States teach a remedial course in mathematics, taken by vast numbers of students who do not pass the examination for the calculus course. This course is often unsuccessful, in that the students who take it often do not graduate. Hence a very flexible computer-based course, designed to discover and remediate students' problems, could make a major impact on university education in this country. This presentation discusses our work in a prototype project that we hope will eventually lead to such a course. The prototype modules concerned trigonometry.

The project followed the production strategies developed for the last 21 years at the Educational Technology Center at the University of California, Irvine. Pedagogical design was carried out by groups of excellent teachers. Programming by undergraduate computer science majors was done in Pascal. The current status of the material will be discussed.

ABSTRACT: Functions, Statistics, and Trigonometry with Computers: Issues in Implementation

Catherine M. Sarther, Sharon L. Senk, University of Chicago School Mathematics Project, 5835 S. Kimbark Ave., Chicago, IL 60637.

Functions, Statistics, and Trigonometry with Computers (FST) is the fifth of six courses in the curriculum designed for grades 7-12 by the University of Chicago School Mathematics Project. It is intended for average college-bound students who have completed an advanced algebra course. Microcomputers equipped with BASIC, function graphers, and statistical packages are used throughout the course to help students visualize functions, explore relations between equations, tables, and graphs, generate and analyze data, simulate experiments, and develop limit concepts. FST also differs from traditional texts by emphasizing realistic applications, continuous review, reading, and open-ended projects.

A 1988-89 formative evaluation with nine teachers and about 300 students found FST to be implementable in typical schools. Classroom observations, meetings with teachers, questionnaires and interviews with teachers and students, and students' tests and projects show that the extent to which FST is implementable is related to the following characteristics of teachers: knowledge of the content; previous experience with the level of student; previous experience with computers; and beliefs about and experience with other innovative pedagogical features.

ABSTRACT: Integrating Computers and Cooperative Learning into the Mathematics Classroom

Barbara Shelly, Westhill High School, Computer Dept. Syracuse, NY 13219; Patricia Tinto, Syracuse University.

This project focuses on using cooperative group strategies and computers as tools within the context of one mathematics class in a public secondary school in upstate New York. The selected content is Course II of the New York State Integrated Sequence which includes formal logic, mathematical systems, Euclidean geometry, and probability. Strategies used are applicable to most secondary mathematics classrooms. The goal of this project is to create, document and analyze an environment where students actively participate in their learning of mathematics.

Failure and passivity persist in many traditional mathematics classrooms. Yet evidence on the impact of cooperative learning strategies on student learning and recent software developments suggest that this need not be so. This project draws on these areas of educational research and development as a basis for creating an environment in which the student participates in the teaching-learning process and in which the computer is used as a natural tool in problem solving. Project software includes *Rocky's Boots, the GeometricSupposer*, and *Geometry.*

The research design employed is that of a case study. Qualitative data is gathered through teacher interviews, student logs, and classroom observations.

The presentation includes examples of lesson plans, videotapes of classroom situations, and a summary.

MODELS AND SIMULATIONS: HOW ONE HIGH SCHOOL BRIDGES THE GAP FROM COMPUTER SCIENCE TO SCIENCE

Michael Haney
Lola Piper
Blair Magnet Program
313 Wayne Avenue
Silver Spring, Maryland 20910

ABSTRACT

Over the past four years, the teachers of the Blair Magnet Program have developed a rich variety of models and simulations which support problem solving in science. These serve the dual purpose of providing students with insights into both when (science) and how (computer science) to use computers. After a brief introduction, students can build powerful models which extend science course boundaries into areas beyond the traditional curriculum. Examples will be demonstrated which use: Logo in Physics for projectile motion; Spread Sheets in Physics to statistically analyze air track and projectile motion data; Stella in Physics to add drag to free fall experiments; True BASIC in Earth Science for Monte Carlo simulations; Pascal in Earth Science to read digitized star data; Stella in Biology for population studies and predator-prey.

Models and simulations provides ways to extend courses to real world problems without the restraints of time and environment. Students soon learn to isolate the important variables and build models around those variables. With a few simple tools, they learn alternative ways of expressing and exploring dynamic relationships thus permitting them to concentrate on the concepts represented. Models and model building are basic themes that cut across all disciplines at all grade levels.

Computer science and science teachers from novice to expert will be interested in the interdisciplinary approaches used by the Blair Magnet staff. They will see demonstrations of applications which were developed for use in day-to-day instruction. During the four years the program has been in existence, these have been refined to useable and effective labs which encourage crossover between disciplines and dialog among teachers.

Participants will receive specific examples of models and simulations used in Physics, Earth Science, and Biology as well as guidelines for devising their own. Handouts and references will help participants to get started or expand the programs in their own school. Teachers who incorporate these ideas will find that computers become valuable tools which support students in their studies in all courses. In science, students are soon able to do larger and more complex problems by extending simple models to include factors which are normally ignored because they are not technically feasible.

DACOTT 21/20
District And Community Of Tomorrow Today

Gary Peterson, Superintendent
Cyndy Everest-Bouch, Project Director
Belridge School District
McKittrick, CA

ABSTRACT

In the summer of 1987, the Belridge School District, a rural K-8 one school district located near Bakersfield, California, posed a critical question. Are we doing enough to prepare children for a future of rapid social and technological change? The governing board realized that education systems of the next century would be different from those of today and determined that to best prepare students for life in the "information age", that current educational practices needed to be transformed, and that one of the main components of this transformation should involve the integration of technology. Throughout the fall of 1987, the board, administration, staff, and community developed a vision of what they wanted their school to become. As one would expect among any group of people, there were varying views and concerns about the direction the change process would take. There was, however, general consensus on what was needed:

- A commitment to maintaining a learning environment that is challenging, exploratory, fun, cooperative, and that draws upon a variety of teaching methods and resources.

- A respect for and reliance upon a teaching staff as key implementors of the educational program and a commitment to provide them with numerous opportunities for professional and personal growth in return for their willingness to trust, accept change, risk, and extend themselves.

- A policy that makes the physical plant, staff, and administration available to the community, welcoming community members both as contributors and learners.

- A reliance on a governing board that embraces change as a fact of life, believes in the DACOTT 21/20 educational vision, and allocates the resources and materials necessary to make that vision a reality.

- An effort to develop and maintain partnerships, both with business and educational groups, who share these beliefs and interests and have made a commitment to improving the education level of our society.

The vision was to combine all of these ingredients with the most effective up-to-date learning tools available and with the support and resources necessary to develop and implement a prototype "Community of Learners" Model Technology Project.

The innovative DACOTT 21/20 project was implemented from the ground up. Teachers, office staff, and families each received an Apple IIgs or Macintosh, plus Imagewriter II printer in the spring of 1988. Software was also provided with each computer. During the remaining school year, the district provided inservice for staff and community to acquaint all with the "friendliness" of their computer. During the summer, extensive physical preparations to the facility and classrooms were completed. In September, 1988, students returned to school to find an Apple IIgs at each of their desks. To insure timely access to the computers by staff and students, all were connected to a local area network which is supported by five Macintosh file servers. Other technological components of the project that were introduced in the fall included: a multimedia center, an applied technology lab, an ESC self-paced learning lab, and an "electronic" music room. The telecommunication component includes: both satellite and ITFS microwave dish reception, modems for each staff member and family household, on-line communication with other schools both nationally and internationally, and an FCC licensed low-power transmitter which allows for UHF TV broadcasting to each home in the district from the school site.

While the DACOTT 21/20 project represents one of the most technologically saturated sites in the country today, our presentation is not about lots of hardware. Rather, it is a session that personalizes technology; emphasizes shared experiences; focuses on people and change environments; and raises some provocative questions as to how we should look at schooling, staff development, and the curriculum.

SPONSOR: SIGCUE

COMPUTERS IN TEACHER ENHANCEMENT

Peggy Healy Stearns
Clifton S. Harris
Harold R. Strang

ABSTRACT: A Comparison of Teacher and Student Evaluations of Educational Software

Peggy Healy Stearns, University at Buffalo, State University of New York.

This research addressed two general questions. First, does training in the evaluation of educational software change teachers' perceptions of this medium? Secondly, should students play a greater role in the evaluation of educational software?

The study compared the perceptions and evaluations of three groups: teachers who had received specific training in evaluating educational software; teachers without such training; and students in the fifth and sixth grades. All groups were approximately matched in terms of computer experience.

Members of each group reviewed two sets of programs: four vocabulary programs and four problem solving programs. They were asked to assign each program a rating of 1 to 10 on six criteria—overall rating, educational value, enjoyment, ease of use, manual, and appropriate use of the computer. Comments were required on overall rating but were optional on other criteria.

Differences in the evaluations of students and teachers were especially apparent in two areas: ease of use and motivation. Ratings of the two groups of teachers differed from each other as well. Trained teachers, for example, were more concerned with an appropriate use of the technology and with the value of the manual and other supplementary materials.

Based on these and other findings, the researcher arrived at the following conclusions: (1) training can make a difference in how teachers perceive and evaluate educational software; and (2) a greater degree of student input should be sought in the software evaluation process.

ABSTRACT: Teacher Workstations: A Key to the Wider Implementation of Technology Throughout the Curriculum

Clifton S. Harris, Associate Professor, The University of Texas at Dallas, P.O. Box 830688, Richardson, TX 75083-0688.

The workstation project has as its major objective the development of a taxonomy of cognitive correlates that link everyday teacher planning and preparation tasks with some capability provided by a computerized workstation. The investigation will enable me to identify the (thinking) steps the teacher takes and the support structures needed in executing the task, as evidenced by screen displays and verbal reports made by the teacher. Such steps and structures will be analyzed and classified into "cognitive correlates" which will then be synthesized into an organized listing known as a taxonomy. The benefits of a taxonomy of this nature are that it will (1) shed light on our understanding of teacher decision-making in required tasks, (2) inform educators and others in how to properly tailor computerized workstations for teachers, and (3) provide needed guidance for preservice and inservice training of teachers in the use of technology. A second product of the research will be a report evaluating the existing technology in the light of the findings of this study.

ABSTRACT: The Curry Teaching Simulation: A Growing Reality

Harold R. Strang, Curry School of Education, University of Virginia, Charlottesville, VA 22903.

This computer-based simulation enables teachers-in-training to practice using fundamental instructional and classroom management skills in an environment which makes behavioral, perceptual, and cognitive demands that are very similar to those made on teachers in real classrooms. During a typical session, the participant conducts a spelling lesson with a class of 16 software-defined pupils projected on a wallscreen via a liquid crystal display projection panel and an overhead projector. Each pupil can emit a variety of authentic speech-synthesized replies to teacher questions—replies that allow for real-time teacher-pupil dialogue exchanges. Pupils can be preprogrammed to exhibit varying levels of lesson preparation or propensity to emit common misbehaviors such as cheating, inattention, out-of-seat, whispering, and noise making. Immediately following a lesson, the participant reviews a series of printed records and screen displays which clearly portray the participant's effectiveness in conducting the lesson. The most recently developed displays allow the participant to compare personal performance (1) with pre-feedback perceptions, (2) with that achieved by peers, and (3) with that achieved by experienced teachers.

AUTHORING TOOLS: PROMISE AND REALITY

David Bantz
Humanities Computing
Dartmouth College
Hanover, NH 03755

Carol Scheftic
CDEC, Carnegie Mellon University

Tom Kurtz
Co-Founder, True Basic, Inc.

Otmar Foelsche
Dartmouth College

ABSTRACT

To what extent can authoring systems allow teachers to design and implement educational computing applications to meet particular instructional needs? Are the tools powerful enough to empower teachers with little or no programming experience to create applications? Can authoring tools provide the sophisticated user interfaces and flexibility that users are coming to expect in microcomputer applications?

Speakers in this session will describe the creation of software in a variety of powerful environments. Each will describe a particular environment or authoring tool and its special strengths, and illustrate its use in producing a useful application. Questions to be addressed include the extent of programming or other computing expertise which is required to use the authoring tool and the amount of effort required for various aspects of creating educational applications, from sketching a design through prototypes to finished applications. Based on experience with these environments, speakers will address the characteristics desired in future enhanced authoring environments.

Carol Scheftic of Carnegie Mellon University's Center for the Design of Educational Computing will describe *cT*, a portable authoring tool derived from the Tutor language which can interactively generate code from drawings and graphs manipulated by the author.

Tom Kurtz, Professor of Mathematics at Dartmouth and Co-Founder of *True Basic*, will describe features of *True Basic*'s implementation of a structured Basic environment which allows the exact same programs to be compiled for use on a wide variety of machines, including MS-DOS and Macintosh.

Otmar Foelsche, Manager of Dartmouth's Language Resource Center will describe the uses of Apple's *HyperCard* for fast prototyping and development of applications, and the creation of a hypertext environment in which texts, graphics and reference materials are integrated into a rich environment for exploration.

FUTURE LEARNING SYSTEMS

Paul Evans
IBM Educational Systems
4111 Northside Parkway
Atlanta, GA 30055

Alfred Bork
University of California, Irvine

Alphonse Buccino
University of Georgia
Athens, GA

Glen Hoptman
Smithsonian Institute
Washington, D.C.

Robbie McClintock
Teachers College, Columbia University
New York, NY

ABSTRACT

The last decade has been a time of transition for computers in edcuation. Limited function computers were used as tools and supplements to the learning process. However, in the next decade, we will begin to use technology to assist teachers and help students by creating very powerful environments for learning—media-rich environments not possible a few years ago.

Scenarios for use of media-rich, technology-centered learning environments will be presented for home, schools, and universities. These environments will encourage linking information; the analysis of information; the arrangement and rearrangement of information; and the presentation of multiple symbolic representations of information (e.g., video, audio, graphic, and text). They will allow alternative pathways through vast databases under the control of the learner or teacher. They will serve not only as intelligent presenters of information, but also as finders, creators, and connectors of information. These databases at first may be subject area specific, but will not be limited and could be continually and constantly be refurbished.

The viability of these scenarios will be discussed relative to: 1) the functionality of emerging digital technologies; 2) the market acceptability and opportunity of wide-scale application and adoption in schools and the home; and 3) the qualitatively new kinds of relationships between teacher and learner that will likely result from their implementation.

THE BUDDY PROJECT: A HOME-SCHOOL PARTNERSHIP

Anita Griffith
Peggy Vogel Hill
Kristen Hein
Indian Creek Elementary
Indianapolis, Indiana

ABSTRACT

Indian Creek Elementary School is a magnet school with a commitment to science and technology. This year we became involved in a pilot program, the Buddy Project. This project began with the implementation of a networked lab. Each fourth grader was provided with a microcomputer and telecommunications terminal in the home. The home computer was supported with curriculum related courseware, a word processing program with graphics capabilities, student management tools, and a printer. The teacher's home computer included the same courseware with the addition of teacher management tools. The students have recently received *Prodigy*, a telecommunications system that links students with the world outside the classroom. The next step will involve the implementation of *Linkway*, a hyper-technology for educators and students. As a result of these innovations our students see technology as a tool for creating a productive future.

Our Discussion will include:

1. Historical overview of the project
2. Intended objectives for the project
3. Implementation of the project at our school:
   Tomorrow's Education Today - video
4. Demonstration of *Linkway*
5. Demonstration of *Prodigy*
6. Projections for year two

SPONSOR: SIGCUE

USER INTERFACES FOR THE ELDERLY: A RESEARCH EXPERIENCE FOR UNDERGRADUATES

Margaret Christensen
College of Information Studies
Drexel University
Philadelphia, PA 19104

Introduction

In one summer project, undergraduate college students helped the elderly, gained professional experience in large scale software development (involving concurrent processes, networks, voice messaging and synthesis, graphics), filed a technical report, and submitted a research paper for conference presentation. Using social worth, student learning, and research productivity as measures of success, this was a particularly successful summer project for undergraduates.

The Project: Eldernet

Eldernet, when fully implemented will comprise a network of computers and the software which drives it (called EASE—the Especially Approachable Software Environment), including a user interface designed for use by elderly people. The network provides two special resources to each member: communication with the community when physical participation has been curtailed due to illness or infirmity; and a medication monitoring service which can be activated either by the member or, in case of the member's incapacitation, by default. In addition, it provides access to the usual software found on computers today (e.g., games, word processing, spreadsheets, etc.) One computer is placed in each member's home and linked via telephone to all other computers on the network; one "nurse's station" is monitored by a medical staff member who responds immediately to emergencies after being notified by the user's computer; and an electronic newspaper is edited and maintained at an "editor's station".

A technological means of helping the elderly to live independently is especially desirable in light of three demographic trends: the increasing average age of the nation's population, the propensity of the federal government toward leaving the care of the elderly to the private sector, and the increase in the number of families in which all adults work outside the home during the day and thus cannot care for elderly family members. Elderly people often receive little support and have few ways of interacting with other community members. However, they have heretofore received scant attention from hardware and software designers. With little published about computer interfaces for the elderly, it seemed possible for undergraduates to contribute to knowledge in the field.

Since many of the elderly users may have no experience using computers, the project required the development of an unusually user-friendly computer interface. From a computer science point of view, Eldernet is of interest as a system that can be used with minimal "training" (like a dishwasher, a microwave oven, or any other household appliance) yet which is cooperatively extensible to a system which the experienced user finds more powerful. A paradigm of object-oriented help based on resolution-based reasoning (Christensen (1986)) is included in the interface, and, where possible, a metaphor of television-telephone communication is used for ease of learning, under the theory that it is easier to learn about a new system by analogy to familiar systems.

The project was chosen by the faculty member first for its social value...so much of a computer science student's time is already spent in writing toy or text book applications or in video games. It appeared that a significant contribution could be made by undergraduates during only one summer, and in addition, most of the resources needed were already available.

Selection

Early in the spring semester potential projects were announced by interested computer scientists at colleges and universities participating in, and funded by, the Pew consortium (Bryn Mawr, Bucknell, Haverford, Lafayette, Muhlenberg, University of Pennsylvania, Princeton, Swarthmore, Vassar). Interested students from those schools then submitted applications for their first and second choices. A faculty committee selected students for the projects based on a perceived fit of qualifications and requirements subject to the constraint that at most one student could stay at a home institution, thereby encouraging communication between institutions. Four students were selected for Eldernet. Qualifications for Eldernet included an ability to program in C, familiarity with UNIX, and preferably course experience in artificial intelligence and/or logic. It happened that the four students who worked on Eldernet had complementary backgrounds: one math major, one computer science major, one English major, and one electrical engineer, all between their sophomore and junior years. In addition, the National Science Foundation supported one highly talented minority high school junior from the South Bronx for eight weeks. This student had spent the previous academic year in college preparatory classes at Strath Haven High School, Wallingford, PA, while living at the ABC (A Better Chance) House in Swarthmore, PA and was recommended by the ABC house parent. For his first four weeks, he worked in the laboratory along side of us, learning UNIX and C by reading textbooks, implementing exercises, and asking the laboratory student system administrator or us for help when he needed it. After that he joined us and was asked to modify and extend code initially written by the college students.

Resources

The Computer Science Program at Swarthmore College fortunately had the core of hardware and software necessary: SUN workstations, UNIX, C, X-windows, micro-EMACS (we particularly wanted to produce portable software). Berkeley Speech Technologies very generously loaned hardware and software providing speech digitization and text to voice synthesis. Swarthmore's Engineering Department loaned a PC clone to house the voice card and software.

Project management

Work on implementing the functionality and the user-friendly interface for Eldernet broke naturally into separate processes:

- one for medication and emergency monitoring;
- one for on-line context sensitive help;
- one for voice messaging and synthesis on the PC;
- and one, which we called the TV, which handled all other applications, especially mail, but including games and a newspaper.

Each student then could look back at his finished work and feel closure as well as being required to see the whole project. Coordination of work done in this fashion is very tricky, from a computer science point of view, from an educational point of view, as well as from an organizational point of view. For example, although we met regularly and discussed the project as a whole, students were reluctant to abandon work on embellishments of "their own" processes in order to test interprocess communication. This organizational trait turned out to be important from a computer science point of view. While getting processes to communicate appeared to be straightforward from documentation and a few small tests, in fact the operating system, SUNOS 4.0, contained a bug in the communication construct (sockets), a bug which did not show up until more extensive testing had been done. For another example, we felt that the graphic design for each process in its own window looked, subjectively, pretty good. However, since the processes were developed by separate people, and not tested simultaneously until late in the summer, even though we tried to design in the large as well, overall, the graphic design needed more cohesiveness. I think that earlier simultaneous testing of prototypes and/or the inclusion of a graphic designer could have prevented a need for some graphic redesign.

On the other hand, from an educational point of view, this provided students with a realistic application involving communicating concurrent processes, something which had not been part of any of the student's backgrounds when they came, and something they might not see outside of the narrow confines of an operating systems course. The issues raised included how procedures would communicate (e.g., private line, reserved memory, blackboard paradigms) and how processes would be synchronized (e.g., send-and-proceed, send-and-wait, wait-to-receive, interrupt driven receive, polling). A hybrid resulted where one main process forked three subprocesses. Of these three subprocesses exactly two communicated with the third. Some data was sent during those communications and all processes also had access to global system data. Students received immediate feedback from erroneous hypotheses of how the communication and synchronization should work, e.g. messages from processes would be incorrectly interspersed, or help would be explaining what had happened five minutes ago. They acquired a "feel" for concurrency that is difficult to transmit through text book examples, and a new appreciation for the difficulty in coordinating cooperative work.

We had 10 weeks of funding, during which the computer was down for 1 week, reminding us of practical, real-world considerations during project management and a need for flexibility in project scheduling. We spent the week doing project design and library research. In addition one week was used for observing initial reactions from some volunteer elderly subjects and for writing and presenting results.

At the end of the summer, a working prototype was built. Iterative design was our motto, and its use was obvious to us. Ideally, we would have begun with a perfect model of our users, a model of our computer system, and a model of the interaction between the two. With little information about elderly users, we began by designing and building a prototype which is to be repeatedly evaluated and improved. The initial design was based on the little data available in the literature, hardware and software availability, and on our own intuition of what would be appropriate. Students learned the importance of bringing the end user into the design process as early as possible. While we were unable to ask elderly users their ideas about an imagined device that was inconceivable to them, we found that they were happy to suggest improvements to an existing prototype that we had designed. From our initial informal feedback from the handful of elderly (and others) who tested Eldernet, we think that type fonts must be larger (greater than 14 point), that we should stream line our screen display, and that we need more information before choosing the "best" voice for text to speech synthesis. The next step is to make these alterations, to test the software further to ensure reliability, and to test Eldernet in a community setting.

It is important to note that while the project was proposed by the faculty member and that resources were obtained by the faculty member, most of the research, design, and implementation was done by the students with guidance from the faculty member when requested.

A look at EASE

EASE is the Especially Accessible Software Environment for Eldernet that was developed by the students. As mentioned above, there are four main processes in that environment. Figures 1 through 3 show the monitor, TV and Advice windows, i.e., those three processes with visual output. The voice process was used by those three processes to speak when wanted.

A click on the panic button in reverse video initiates an automatic phone call and panic message transmission to the nurse's computer which must be near a nurse at all times. The nurse then follows up with a telephone call. If there is no answer or if the nurse deems it necessary s/he provides emergency aid. The other buttons are used for operations on a person's schedule for medications. According to that schedule, an announcement replaces the buttons in this area stating that it is time for medication, and the exact dose of the medication(s). Depending on the nurse's choice, the nurse can be summoned if the person does not signify (by clicking on a box) that the medication has been taken.

In Figure 2, the user has clicked on the "mail channel" and the mail application has appeared on the simulated television screen. When the flag is up on the mailbox icon, there is mail to read. Clicking on boxes lets the user choose voice or text mail. A customized version of micro-EMACS is used for input of text mail.

In Figure 3, clicking on the round buttons provides solicited advice while unsolicited positive feedback is continuously offered from applications in the monitor and TV processes. For example, if the user is not sure if the right button was pushed to mail a letter, this process will tell what has happened. If the user want to learn how to do something, the user requests solicited advice by clicking on the appropriate button. This computer help system (called advice here so as not to be confused with emergency help) is generating explanations based on the specifications of the software in the system.

All emergency notices are signified with sound. The user can also select which of the other text the user wishes to hear spoken.

Conclusion

It is not easy to find a means to offer students professional experience in large scale software development, or research opportunities, or a means to use computers to directly benefit others in the community. Eldernet did that and is offered here as a small file card in one's hyperstack of ideas to prick one's own imagination.

Acknowledgments

This project would not have been possible without support from the Pew Foundation and from the National Science Foundation (supplement to Grant NCR-8702546). Furthermore, it should be clear to those reading the paper that the project depended upon the outstanding work of Robert Gottshall (Bucknell Univ.), Jed Hartman (Swarthmore College), Sharad Chaudhary (Lafayette College), Doug Yatcill (Bucknell Univ.), and Dean Anton (who is now living at the ABC House in Swarthmore, PA, while attending Strath Haven High School). It also depended upon Swarthmore College's making computer facilities available during the summer. Many thanks are due to Clif Kussmaul for the loan of a Microsoft C compiler from Swarthmore's Engineering department. We also greatly appreciate the help given us throughout the project by Hal Pomeranz, the SUN student system administrator at Swarthmore. Overall, it was a very cooperative group who made this project possible...thank you, and please keep that spirit.

References

Baecker, R.M. and Buxton, W.A.S. (eds.), *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1987.

Baker, S. M., "X Windows--A View from DOS," *Programmer's Journal*, July/August (sic.) 1988.

Bowe, F., "Making Computers Accessible to Disabled People," in Baecker, R.M. and Buxton, W.A.S. (eds.), *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Los Altos, CA: Morgan Kaufmann Publishers, Inc., pp.709-713, 1987. (Reprinted from *Technology Review* 90(1):52-59.)

Card, S.K., Moran, T.P. and Newell, A., *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

Christensen, M., *Explanation Generation from Algebraic Specification Through Hyperresolution and Demodulation: Automated Heuristic Assistance*, Doctoral Thesis, Philadelphia: Temple University, 1986.

Efe, K. and Miller, C., "The Kiwinet-Nicola Approach: Response Generation in a User-Friendly Interface," *Computer* 16(9):66-78, 1983.

Fenchel, R. and Estrin, G., "Self-Describing Syst ems Using Integral Help," *IEEE Transactions on Systems, Man, and Cybernetics* 12(2):162-167, 1982.

Finen, T., "Providing Help and Advice in Task Oriented Systems," in *Proceedings of IJCAI, Volume 1*, Los Altos, CA: William Kaufmann, 1983.

Foley, J., "Guest Editor's Introduction: Special Issue on User Interface Software," *ACM Transactions on Graphics* 5(2):75-78, 1986.

Genesereth, M., "An Automated Consultant for MACSYMA," in *Proceedings of IJCAI*, Los Altos, CA: William Kaufmann, 1977.

Genesereth, M., *Automated Consultation for Complex Computer Systems*, Doctoral Thesis, Cambridge, MA: Harvard University, 1978.

Gettys, J., Newman, R. and Della Fera, T., *Xlib—C Language X Interface Protocol Version 10.*

Gould, J.D. and Boies, S.J., "Speech Filing—An Office System for Principals," in Baecker, R.M. and Buxton, W.A.S. (eds.), *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Los Altos, CA: Morgan Kaufmann Publishers, Inc., pp. 8-24, 1987. (Reprinted from *IBM Systems Journal* 23(1):65-81, 1984.)

Gould, J.D. and Boies, S.J., "Humaan Factor Challenges in Creating a Principal Support Office System," in Baecker, R.M. and Buxton, W.A.S. (eds.), *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Los Altos, CA: Morgan Kaufmann Publishers, Inc., pp.25-37, 1987. (Reprinted from *ACM Transactions on Office Information Systems* 1(4):273-298, 1983)

Grossner, C.P., Radhakrishnan, T. and Pospiech, A., "An Integrated Workstation for the Visually Handicapped," *IEEE Micro* 3(3):8-16, June 1983.

Guttag, J., "Abstract Data Types and the Development of Data Structures," *Communications of the ACM* 20(6):396-404, 1977.

Hartson, H.R. (ed.), *Advances in Human-Computer Interaction, Volume 1*, Norwood, NJ: Ablex Publishing Corporation, 1985.

Hiltz, S.R., *Online Communities: A Case Study of the Office of the Future*, Norwood, NJ: Ablex Publishing Corporation, 1984.

Jacob, R.J.K., "A Specification Language for Direct-Manipulation User Interfaces," *ACM Transcations on Graphics* 5(4):283-317, 1986.

Ledgard, H., Singer, A., and Whiteside, J., *Lecture Notes in Computer Science 103: Directions in Human Factors for Interactive Systems*, Berlin: Springer-Verlag, 1981.

Mark, W., "Representation and Inference in the Consul System," in *Proceedings of IJCAI*, Los Altos, CA: William Kaufmann, 1981.

Mitchell, C.M. and Forren, M.G., "Multimodal User Input to Supervisory Control Systems: Voice Augmented Keyboard," *IEEE Transactions on Systems, Man, and Cybernetics* 17(4):594-607, 1987.

Monk, A. (ed.), *Fundamentals of Human-Computer Interaction*, London: Academic Press, 1984.

Nickerson, R.S., *Using Computers: Human Factors in Information Systems*, Cambridge, MA: The MIT Press, 1986.

Rissland, E.L., "Ingredients of Intelligent User Interfaces," in Baecker, R.M. and Buxton, W.A.S. (eds.), *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, Los Altos, CA: Morgan Kaufmann Publishers, Inc., pp.703-708, 1987. (Reprinted from *International Journal of Man Machine Studies* 21:377-388, 1984)

Rothwell, R. and Robertson, A.R., "The Contribution of Poor Communication to Innovative Failure," *Aslib Proceedings* 27(10):393-400, 1975.

Schank, R. *Conceptual Information Processing*, New York: American Elsevier, 1975.

Scheifler, R.W. and Gettys, J., "The X Window System," *ACM Transactions on Graphics* 5(2):79-109, 1986.

Schneiderman, B., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Reading, MA: Addison-Wesley Publishing Company, p.vi, 1987.

Simpson, H., *Design of User-Friendly Programs for Small Computers*, New York: McGraw-Hill Book Company, 1985.

Sun Microsystems, *SunOS™ Reference Manual*, pp.189-290, 1988.

Swartout, W., "Producing Explanations and Justifications of Expert Consulting Programs," Technical Report, Department of Electrical Engineering and Computer Science, Cambridge, MA: MIT, 1980.

Swartout, W., "The GIST Behavior Explainer ," ISI/RS-83-3, Marina Del Rey, CA: ISI/USC, 1983.

Wilensky, R., Arens, Y., and Chin, D., "Talking to UNIX in English: An Overview of UC," *Communications of the ACM* 27(6):575-593, 1984.

Wos, L., Overbeek, E., Lusk, and Boyle, J., *Automated Reasoning*, Englewood Cliffs, NJ: Prentice-Hall, 1984.

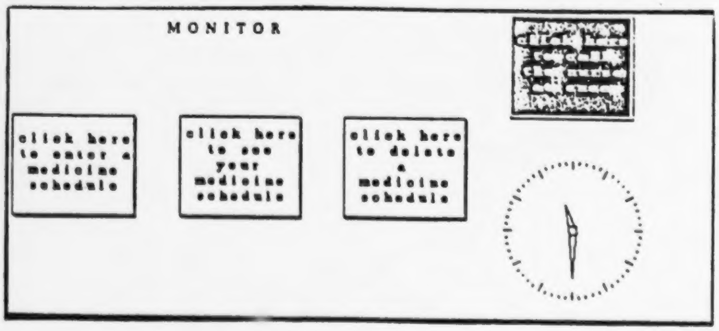

MONITOR
click here to enter a medicine schedule
click here to see your medicine schedule
click here to delete a medicine schedule

Figure 1 The monitor process

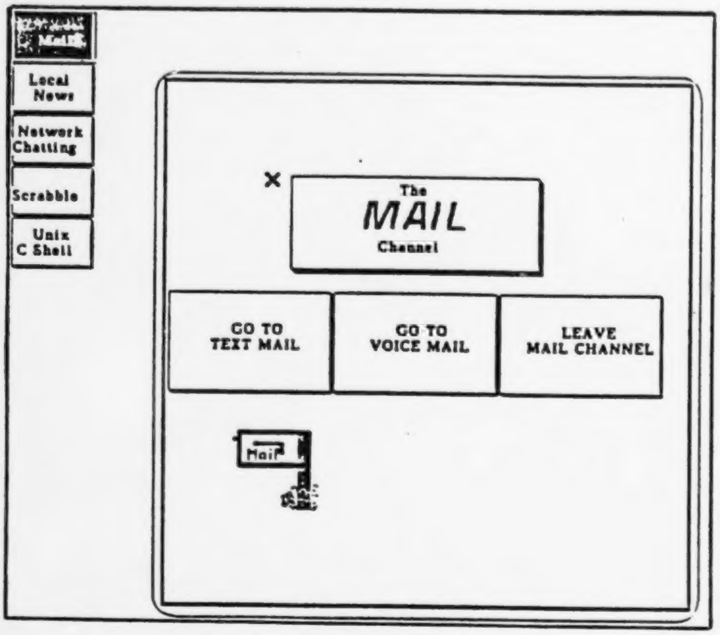
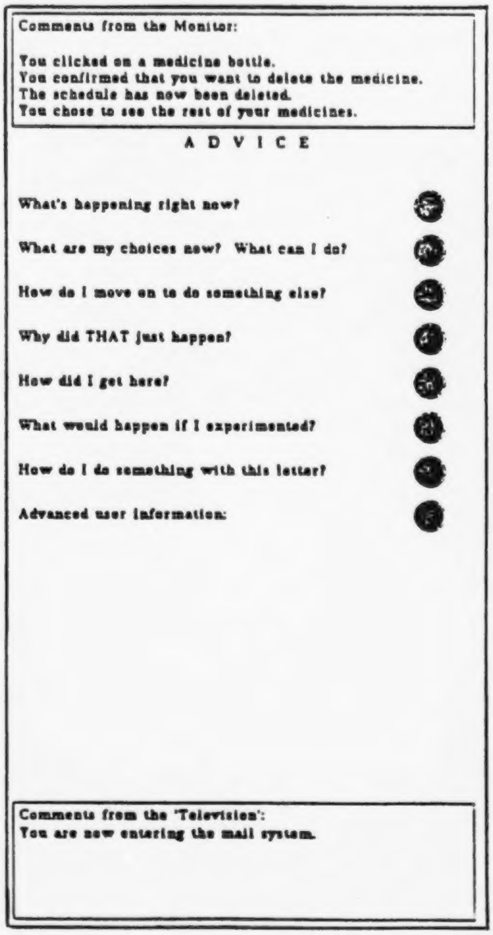

Local News
Network Chatting
Scrabble
Unix C Shell
×
The MAIL Channel
GO TO TEXT MAIL
GO TO VOICE MAIL
LEAVE MAIL CHANNEL
Mail

Figure 2. The TV process



Comments from the Monitor:

You clicked on a medicine bottle.
You confirmed that you want to delete the medicine.
The schedule has now been deleted.
You chose to see the rest of your medicines.

ADVICE

What's happening right now?
What are my choices now? What can I do?
How do I move on to do something else?
Why did THAT just happen?
How did I get here?
What would happen if I experimented?
How do I do something with this letter?
Advanced user information:

Comments from the 'Television':
You are now entering the mail system.

Figure 3. The advice process

FACULTY-UNDERGRADUATE PARTNERSHIPS IN COMPUTER SCIENCE RESEARCH

Marc J. Lipman and Robert L. Sedlmeyer
Indiana University - Purdue University at Fort Wayne
Computer Science Department
2101 E. Coliseum Boulevard
Fort Wayne, IN 46805

Abstract

Indiana University - Purdue University at Fort Wayne (IPFW) is an urban, primarily undergraduate institution, whose primary mission is education. IPFW's objective is to give undergraduates not only foundational knowledge, but also the learning skills necessary to succeed throughout their professional careers. We believe that one way to do this is to create faculty-undergraduate research partnerships. The GRAPPLE project provides opportunities to form these partnerships. We manage the project in a way that permits students to work closely with faculty. Since the inception of the GRAPPLE project in 1984 we have sponsored fourteen undergraduates, an average of two per semester. Some students participated for as many as four semesters. In this paper we report our experiences with these partnerships, their creation, managment, costs, and benefits.

Introduction

Indiana University - Purdue University at Fort Wayne (IPFW) is an urban, primarily undergraduate institution, whose primary mission is education. That does not excuse the faculty from reserach responsibilities, it just means that our research resources are severely limited and our teaching loads are high.

IPFW has created a task force to study the "Nature and Quality of the Undergraduate Experience." One way to enhance that experience is to incorporate extraordinary educational activities that go beyond the traditional classroom or prefabricated laboratory work. IPFW's objective is to give undergraduates not only foundational knowledge, but also the learning skills necessary to succeed throughout their professional careers. We believe that one way to do this is to have students do research. Doing research allows undergraduates a glimpse beyond the limited horizon of a four-year college education. Research, by its open-ended nature, forces both novel thinking and integration of knowledge from possibly diverse disciplines. Doing research includes disseminating the results: Writing papers and giving talks. It includes developing the ability to function in a team, to tolerate the idea of starting over, to take an active role in creating the questions to be asked, and to explore in greater depth topics covered in ordinary coursework.

We both accomplish our research goals and achieve our expanded educational goals by forming faculty - undergraduate partnerships. A *partnership*, according to the dictionary, is formed by two or more parties who share a joint interest. In our case, the parties are us and our computer science and mathematics undergraduate majors, and the joint interest is the development of the software system we call GRAPPLE. In this paper we report our experiences with these partnerships over the last four years, their creation, managment, costs, and benefits.

The Research Project

GRAPPLE

In 1984 we embarked on a long-term project to develop an integrated set of knowledge-based tools for supporting graph theory research. The project, named GRAPPLE, was divided into four phases: (1) Creation of an interactive graph editor; (2) Development and implementation of algorithms for common graphical properties and operations; (3) Automation of the investigation of user-specified research problems involving graph families, properties and operations; and (4) Modeling the formulation of the research problems and investigation strategies. Each phase builds upon the preceding phase, and the associated tools incorporate increasingly greater knowledge of graph theory.

Over the last four years our efforts have been devoted to the first two phases. We now have a fairly robust system which includes a graph editor, librarian, property computation subsystem, operations subsystem, a menu-oriented user interface, and a fully developed help subsystem.

The editor allows the user to interactively create and modify graphs. Basic functions include adding and deleting vertices and edges, and specifying vertex labels. The user is able to choose background, vertex and edge colors, and whether the labels are displayed. The user can alter the picture of the graph without changing its internal representation: Individual vertices may be moved along with incident edges; windowing allows the user to view a portion of the graph close-up. The editor will handle graphs on up to one hundred vertices.

The librarian manages the storage and retrieval of graphs. This includes both a library of standard graphs and graphs constructed by the user. The standard library includes classes of well-known graphs, such as cycles and paths, and graphs of specific interest, such as the Petersen, Coxeter, and Tutte graphs. The user may incorporate graphs from either library during an editing session.

The properties subsystem computes specified properties of a given graph. These include properties which are basic to the study of graph theory, such as maximum degree, vertex connectivity, shortest path and the number of cliques in a graph. Whenever possible the results of a computation are displayed on the drawing of the graph. For example, the shortest from vertex X to vertex Y can be listed vertex by vertex; alternatively, the appropriate set of edges can be highlighted on the graph display.

The operations subsystem facilitates the construction of complex graphs based on previous graphs. The subsystem includes both unary operations, such as complement, k^{th}-power, and k^{th} completion, and binary operations, such as cartesian product and join.

Further details of GRAPPLE can be found in [SL85, SL86, SL87a, SL87b, SL88].

The Research Environment

To support the development of GRAPPLE we have created an undergraduate computer science research laboratory. This laboratory contains two IBM-AT computers each with 3M memory, 40M hard disk drive, two floppy drives, an EGA display and a Microsoft mouse. Peripherals include an HP7475A plotter and an HP Laserjet+ printer. Program development software includes the Turbo Pascal and Gold Hill Common LISP programming environments and Index Technology's Excelerator. WordPerfect, Ventura Desktop Publisher, and Harvard Presentation Manager are available for preparation of external documentation and presentations. The IBM-ATs, printer, and LISP programming environment were obtained through a grant from IPFW; Excelerator was obtained through Index Technology's Academic Grants Program; the remaining laboratory components were obtained through grants from the Indiana State Vocational Board for Training and Education.

How the project works

The faculty and the undergraduates who work on GRAPPLE have two distinct sets of goals. As faculty, we want this work to simultaneously educate our students, in ways they may not even be aware of, and to provide us with an outlet for our research energies. We want our students to develop the ability to integrate subject matter from several courses to a single, large, ongoing project. We want them to learn how to function in a research team. We want them to become self- motivated learners. On the other hand, a typical undergraduate enters the GRAPPLE environment with only the thought of obtaining credits for work done, perhaps in a setting less confining than the usual classroom. One of our jobs is to make them aware that the skills they acquire while working on GRAPPLE are applicable beyond their undergraduate experience, indeed are precisely those necessary to succeed in graduate school or an industrial research setting. Once they realize that, the nature of the partnership changes, and they become more active in defining the research goals.

The GRAPPLE project provides an ideal vehicle for accomplishing all our goals. We manage the project in a way that provides opportunities for our students to work closely with us.

One of these parnerships is formed every time we agree to sponser a student to work on GRAPPLE. Formally, students enroll in a specific independent study course which we direct. A student selects from one to three credit hours, and we assign tasks commensurate with the number of credits the student is taking. Most students select three credit hours which amounts to approximately ten hours per week of research work. The course is limited to juniors and seniors who have a B+ average (3.2 GPA, A=4), and who have completed the first two years of the computer science curriculum. Many students choose to participate in the project for more than one semester, up to a maximum of six credit hours.

The different responsibilities of the partners parallel the traditional division of management and labor and can be summarized in table 1 below.

Faculty responsibilities	Student responsibilities
Provide overall management	Provide labor
Define research objectives	Achieve the objectives
Obtain resources	Manage resources
Evaluate the work	Provide feedback
Perform basic research	Perform applied research

Table 1: GRAPPLE lines of responsibility

At the beginning of each semester we select the tasks, set up a schedule and assign students to tasks. Students operate in teams of two students each, each team with a particular task. The teams have weekly meetings with us during which they report progress and outline work for the coming week. Teams also make monthly presentations to the entire GRAPPLE staff. Specific tasks can be restructured to accommodate overly optimistic judgments of student productivity or to take advantage of serendipity.

Every team is required to write a report and make an oral presentation of their semester's research. These reports are prepared for joint publication as appropriate, and the students are encouraged to present them at the annual local Sigma Xi research competition. Three things determine a student's grade: 1) The degree of completion of the team's assigned task, 2) the contribution of the student to the team's effort, and 3) the quality of the written and oral presentations.

Initially students are given well-defined tasks, such as the implementation of a known algorithm. As their skills mature the tasks change to more ill-defined ones, such as deciding how a graph should be drawn on the screen to be of most use to the graph theorist. Students are encouraged to voice their views as to how the work should proceed. We lead by defining their work, critiquing their efforts, and augmenting their knowledge of the subject matter through impromptu discussions and directed readings of the technical literature. Continuing students are given more flexibility in terms of defining project objectives, schedule and team management.

Since the inception of the GRAPPLE project we have sponsored an average of two undergraduates every semester, including summers. Some students participated for as many as four semesters. In the development of the first two phases of GRAPPLE we worked with fourteen undergraduate. As of August 1988, four are completing their bachelors' degrees, one is in graduate school, another is in the process of applying to graduate school, and the rest have graduated and are computer professionals.

Costs and Benefits

Working on the GRAPPLE Project forces a student to integrate knowledge acquired from diverse sources, be it coursework, technical literature, or our impromptu, topical lectures. The work exposes undergraduates to the creative aspects of research, including both the frustrations and the successes. We actively encourage our better students to continue their education. Because of their opportunities to interact closely, one-on-one, with faculty, they become more open, more willing to make mistakes, and perhaps more eager to please us. Because the results of their work have importance beyond the classroom, our students discover that working in the research environment generates a great deal of enthusiasm for the work in its own right.

For the faculty the most obvious benefit of this type of partnership is an increase in productivity. As every research faculty member at a Ph.D. granting institution knows, the presence of graduate students increases the amount of work generated. For us, these undergraduates serve the same purpose. They keep our energies flowing, and produce results that dovetail nicely with our own to produce more publications.

A less tangible benefit for us is the gratification that comes from the mentoring relationship itself. It is immensely enjoyable to teach these good students. We feel that we have a great impact on them, that we have the opportunity to teach by example. Furthermore, their presence and energy invigorates us: When a student is interested in what we teach, our interest in teaching is elevated.

Forming and managing successful partnerships requires ongoing committment from the faculty. Directing an independent study course must be done on an overload basis; and, since each partnership is individually tailored to meet the needs of the project and the student this commitment goes beyond a single, additional perparation. As the project has grown there have been increasingly greater training costs involved in familiarizing students with GRAPPLE so that their work may be properly integrated. While this is a problem with any large, long-lived software product it is exacerbated in GRAPPLE since project personnel turnover almost every semester. Many tasks require knowledge drawn from artificial intelligence, software engineering, and computer graphics: Good students are a must. Recruitment of these students is a demanding and never-ending task. Finally, participation in GRAPPLE fulfills elective requirements in the BS degree. By taking as many as six hours a student may miss other elective courses (such as data base management) that enhance educational breadth.

Conclusion

GRAPPLE development continues. Our current research focuses on two distinct types of problems: 1) Devising and implementing algorithms for important graph theoretical parameters; and 2) Representing graph theoretic knowledge in forms suitable for reasoning about graphs and their properties. Each of these broad areas contains many tasks suitable for faculty-undergraduate partnerships in research.

These partnerships work. The GRAPPLE system, which contains approximately 30,000 lines of code, provides a useful research tool. Even in its unfinished state it has been distributed to nine universities. We have six publications directly related to GRAPPLE, and have made more than that number of presentations. Fourteen students have participated, nine of them for more than one semester. Former students have told us that their GRAPPLE experience was useful in their later jobs.

Even so, we have to compete for the attention and interest of students, and advanced computer science students don't have difficulties finding good summer employment. Therefore, we are searching for external funds to allow us to hire our students to continue to do GRAPPLE work in the summer.

References

[SL85] M.J. Lipman and R.L. Sedlmeyer "Computer-Aided Investigation in Graph Theory [Abstract]," *Proceedings of the 1985 ACM Computer Science Conference* (1985), p 418.

[SL86] M.J. Lipman and R.L. Sedlmeyer "Developments in an Expert System for Graph Theory Investigation," *Proceedings of the 1986 ACM Computer Science Conference* (1986), pp 327 - 330.

[SL87a] M.J. Lipman and R.L. Sedlmeyer "Property Computation in Grapple [Abstract]," *Proceedings of the 1987 ACM Computer Science Conference* (1987), p 366.

[SL87b] M.J. Lipman and R.L. Sedlmeyer "Knowledge and Reasoning in Graph Theory Research [Abstract]," *Proceedings of the 1987 ACM Computer Science Conference* (1987), p 415.

[SL88] M.J. Lipman and R.L. Sedlmeyer "Experimental Graph Theory with a Microcomputer," *Proceedings of the Fourth Academic Microcomputing Conference*, 1988.

COMPUTERS IN HIGHER EDUCATION

William E. Grimsley, Jr.
Robert A. Gray

ABSTRACT: Connecting Collegiate Curriculum Computing

William E. Grimsley, Jr., Chairman-Computer Information Systems Department, Director-Graduate Program in Educational Computing, The University of Charleston, 2300 MacCorkle Avenue, S.E., Charleston, WV 25304.

C-CUE (Consortium for Computing in Undergraduate Education) has been in existence for only two years following a three year development process. It is committed to the exploration and advancement of appropriate uses of computing and other information technologies in undergraduate education.

This presentation will provide information pertaining to the involvement of a group of small, liberal arts colleges in the improvement of computing across the curriculum. It is not about the connection of hardware between institutions of higher education. It is not about the sharing of a large mainframe computer at a centralized location by several colleges and universities. It is not about the electronic and telecommunication links between several localities. It is not about an organization primarily for the computer science or management information systems departments. It is about the sharing of information, ideas, problems, solutions, hardware acquisition, software utilization, and human resources at several institutions that are interested in assisting one another and promoting academic computing in the various disciplines within the curriculum.

Discussion will pertain to the development of such an organization, the components mandatory for stability and growth, the necessity for limitations on institution size and location, and the benefits received by participating institutions.

ABSTRACT: The Development of a Student-Based Educational Software Selection and Evalution Center in Higher Education

Robert A. Gray, Associate Professor of Audiovisual Communications, Department of Audiovisual Communications, Kutztown University, Kutztown, PA 19530.

Initially the paper will focus on the rationale for the Center for Software Selection and Evaluation at East Texas State University in Commerce, Texas. The second part will examine the major steps in developing the program. This will include initial contact with industry, strategies used to obtain software, students involvement, use of rating forms, data analysis of evaluative information, and final report generation. Specific guidelines for the development of a software selection and evaluation center will be examined.

The Center for Software Selection and Evaluation at East Texas State University was developed to provide students in the graduate programs with a quality experience in reviewing and evaluating current educational software. In addition, software producers received maximum exposure for their programs with the future leaders in the field of instructional technology and educational computing. They also received a review of their CAI packages which could be used for advertising purposes and for future revisions. The Center for Software Selection and Evaluation provided mutually beneficial services for students, faculty, and software producers.

MAKING LEARNING VISIBLE: A HI-TECH, LO-TECH, AND NO-TECH APPROACH

Hillel Weintraub
Harvard University Graduate School of Education
78 Coolidge Street
Brookline, MA 01246

ABSTRACT

This presentation would give teachers some ideas about and experience with creating and sustaining a learning environment in which the main focus is helping those involved (students and teachers both) develop an increasing awareness of the environment and their role in it, through the use of collaboration, constructionism, and communication. The essential approach will be that of a short workshop; those attending will participate in a number of exercises designed to explore various ideas about teacher and student roles, evaluation, curriculum, problem solving, and helping behaviors. The central focus will be on exposing as much as we can about learning processes and learning behaviours.

THE CHALLENGES OF INTEGRATING TECHNOLOGY INTO THE CURRICULUM: TEN TRENDS FOR LONG-RANGE PLANNING

Patricia A. Sturdivant
Executive Director for Technology
Houston Independent School District
5300 San Felipe
Houston, TX 77056

ABSTRACT

Seven years ago Houston ISD established a centralized department to coordinate its technology efforts. This session will focus on some of the innovative programs we've tried, the staff development strategies that have been successfully designed to support curriculum integration, and the lessons we've learned in trying to create a district-wide vision for change. Ten important technology and education trends (with implications for long-range planners) will be presented with an eye toward how to develop contingency plans for an uncertain future.

Only when you take,risks do you learn what you can do to improve. Houston ISD has been willing to take calculated risks in launching innovations that some said would never work. These include programs like *Computers Can*, a home computer check-out program, *CompuBuy*, a software library for parents, *Techmobile*, a traveling technology lab, and *E-Net*, a district-wide telecommunications network. Most of these programs were first implemented on a small scale and then expanded across the district. Some efforts have been extremely well received, like the district's development and implementation (230 schools) of its own reading/laguage arts software (*FunReader*, 38 disks), while others have faltered. For example, technology magnet schools (200+ computers per campus) have captured the enthusiastic interest of students, but finding and retaining qualified staff and updating obsolete hardware have been formidable obstacles in implementing these computer intensive programs.

Houston has launched new technology initiatives that support the curriculum and administration through both top-down and bottom-up planning strategies. The learning curve has increased dramatically as more teachers become trained in using technology in their classrooms. (Houston has an in-house team of trainers who provide a 296-hour training program for teachers.) As staff sophistication increases, teachers want to create their own plans for using technology and are less interested in receiving "structured configurations" from the central office. Labs continue to provide for greater efficiency, but many teachers want access to computers in their own classrooms, particularly to improve their own productivity.

With approximately 13,000 microcomputers now in use, it is becoming increasingly more challenging to determine which software should be purchased and how best to support it. Even for a large urban district, it's difficult to evaluate and disseminate the many programs that continue to flood the market. A wide variety of incentives has been developed so that adminstrators and teachers can help review the programs; the district wants to ensure that teachers are using "only the best." There is still much work to be done.

Houston's technology staff includes experts in strategic planning, implementation support, courseware development, hardware procurement, software evaluation, computer maintenance, telecommunications/networking, and video production. Yet, despite this array of experts, the district faces the difficult prospect of planning for dozens of contingencies that may or may not materialize—all in an uncertain budget climate.

These particular technology trends are worth tracking as they impact planning efforts. They must always be juxtaposed with important educational trends that are likely to create new opportunities for using interactive learning devices. This presentation will expound on these ten trends:

- Networking: determining the cost benefits baselines that will justify their use.
- Optical disc-based software: sorting out the formats—CDI, CD-ROM XA, ADPCM, and CDV.
- Computer manipulated speech: exploiting the potential and surviving the compatibility headaches.
- Telecommunications: developing applications that aren't expensive boondoggles.
- Bus wars: procuring hardware and fighting the retrenchment blues.
- Integrated learning systems: juggling system selection and accomodating accountability presures.
- "Restructuring" reform movement: saying we are when we really might not be.
- Family support programs: determining how and what technology is appropriate.
- TASP-type testing: breathing life into competency-based education with computer support.
- Curricular efforts to promote collaborative learning: using interactive devices to ensure skills of cooperation, compromise, and decision-making.

AUTHOR INDEX

Abramson, G 113
Adamson, E 18, 20
Adler, W 294
Ainsa, T 144
Alexander, D 233
Alifrangis, C 289
Alo', R 295
Amos, N 292
Anderson, R 143
Anderson-Inman, L 117, 263, 294
Antenore, F 375
Arnal, G 211
Asen, S 118
Augliere, R 123
Austing, R 40
Baer, V 159
Baird, B 333
Ballew, D 372
Bantz, D 379
Barba, R 212
Barclay, T 117
Bardige, A 230
Barker, K 59, 292
Barnes, B 59
Bathke, N 285
Batson, T 222
Battista, M 209
Beals, D 126
Bear, G 40
Beardslee, E 22
Beaver, J 240
Becker, H 19
Beichner, R 295
Beidler, J 40
Beishuizen, J 332
Bengtssen, B 267
Bernstein, D 268
Bickford, L 300
Biegel, S 293
Biehler, L 283
Biglar, H 334
Billings, K 220
Bitter, G 351
Blankenbaker, R 351
Bloede, M 231
Blurton, C 351
Boone, R 141
Bork, A 375, 380
Brett, G 222
Bright, G 10, 126
Brittain, D 146
Brownell, B 289
Bruce, B 265
Brumbach, E 365
Bruner, C 248
Buccino, A 380
Buchanan, P 231
Bull, G 18
Buzzeo, R 141
Campbell, P 144
Cannon, R 222
Cardello, C 292
Carey, D 286, 333
Carver, S 1
Casey, P 273
Cassel, L 40
Chaiklin, S 227
Chazan, D 54
Chow, D 337
Christensen, M 382
Clements, D 209
Clifford, J 118
Collis, B 264
Connors, P 231
Copen, P 220
Corning, N 122
Corwin, R 187, 312
Cory, S 234
Cossey, D 39
Costa, M 122
Cowan, H 118
Craig, B 209
Cron, M 117, 294
Cummins, D 94
Cunniff, N 147, 273
Darrise, L 374
Dede, C 128
Delclos, V 224
DiGiammarino, F 313
Donahue, B 324
Downing, L 227
Doyen, S 175
Dubitsky, B 141
Duckel, G 287
Dunlap, C 300
Dwyer, C 291
Dwyer, D 104, 223
East, P 373
Edwards, C 227
Ellis, J 262
Evans, P 380
Evans, W 282
Everest-Bouch, C 377
Farr, B 224
Finkel, L 234
Fisher, F 366
Fletcher, G 234, 365
Fletcher, J 282
Foelsche, O 379
Fonesca de Pacheco, C 350
Franklin, C 224
Franklin, S 375
Frederiksen, J 362
Freyd, P 129
Friske, J 221
Funk, J 289
Futhey, T 163
Garcia, L 366
Gardner, M 38, 118
Gau, G 33
Gersting, J 372
Gilbert, S 366
Goldenberg, E P 87
Goldenson, D 194
Goldsmith, M 300
Goodson, B 234
Gorgone, J 282
Gormley, P 40
Grady, A 154
Grady, D 117
Graeber, J 267
Gray, R 390
Griffith, A 381
Grimsley, W 390
Guzdial, M 204
Halapin, J 122
Hall, W 345
Hammon, A 300
Hanchey, C 282
Haney, M 376
Harel, I 72
Harris, C 378
Harris, J 18, 297
Harris, T 144
Harvey, B 97
Hatfield, M 351
Havhgolm, P 356
Hawkins, J 248
Haytock, B 372
Hein, K 381
Heller, S 220, 267
Higgins, K 141
Hill, P 381
Hillinger, M 117, 294
Hogan, S 146
Honey, M 248
Hooper, P 283
Hoptman, G 380
Horwitz, P 262
Houde, R 54
Hunter, B 332
Inbal, E 127
Iovine, M 374
Jackson, I 283

Jacobs, J 126
Jahn, B 261
Johnson, R 125
Johnson, V 125
Johnston, J 120, 228
Jones, M 334
Jongejan, T 71
Jukes, I 335
Kafai, Y 46
Kahn, J 129
Kapisovsky, P 312
Kaput, J 284
Kelemanik, G 363
Kelly, D 38
Kiernan, G 41
Kim, J 337
Kirkpatrick, S 334
Knox-Quinn, C 209
Koetke, W 117
Koltun, A 41
Koontz, T 20
Kozberg, G 374
Kozma, R 120
Kurland, M 117, 122, 230
Kurshan, B 220
Kurtz, D 231
Kurtz, T 379
La Duke, W 146
Land, M 141
Lawson, J 333
Lemke, C 365
Lenk, C 232, 262
Lewis, H 292
Liao, Y 10
Liden, L 373
Lidtke, D 59
Lindroth, L 118
Link, C 211
Lintner, M 221
Lipman, W 387
Lipsitz, L 37
Lovitt, T 141
Lowd, B 313
MacDonald, J 214
MacGregor, K 127
Mallory, B 313
Mandell, S 143
Marcus, S 210
Martin, D 267
Mathias, M 291
Mathies, B 221
Maurer, W 318
Maury, K 231
McCarthy, J 285
McClintock, R 380
McCracken, D 143
McDermott, H 365
McGinnis, M 288
McInerney, C 143
McKavanagh, C 235
McKinley, K 211
McKinnon, D 226
McWhinnie, H 233
Meadows, J 332
Meltzer, L 58
Meshot,C 22
Migliarini, G 313
Miller, I 245
Mims, T 334
Mitrani, M 24
Mlynarczyk, C 221
Mohl, B 296
Mokros, J 312
Molnar, A 128
Moore, P 221
Morgan, T 373
Morrison, D 293
Moss, K 123
Muir, M 334
Naiman, A 245
Nanny, M 296
Neill, M 365
Nelson, C 22
Newbold, W 352
Niess, M 301
Nolan, P 226
Ocko, S 374
Olson, R 294
Orcutt, D 313
Papert, S 350
Patten, B 335
Pearlman, B 349
Pease, P 211
Perrolle, J 143
Perry, M 105
Peterson, G 377
Pinto, P 375
Piper, L 376
Polka, B 291
Potocny, P 363
Psotka, J 128
Quinsaat, M 122
Radford, T 363
Rawitsch, D 224, 349
Rehfield, K 233
Reinhardt, M 187
Reiser, B 362
Repman, J 127
Resnick, M 117, 374
Riel, M 250
Roditi, B 58
Ronkin, J 283
Rosebery, A 265
Rosegrant, T 267
Roskow, A 249
Rossano, K 146
Rubin, A 140
Rucker, C 293
Rush, M 231
Russell, S 140, 312
Sadler, P 364
Saga, H 60
Sales, G 125
Samson, J 123
Sanders, J 288
Sarther, C 375
Saterfiel, T 211
Sayers, D 313
Scali, N 23
Schaffer, M 283, 350
Scheftic, C 163, 379
Schilling, S 117
Schrum, L 220, 232
Schulz-Zander, R 267
Schwartz, E 41
Scrogan, L 232
Sedlmeyer, R 387
Semenov, A 220, 350
Senk, S 375
Shail, G 126
Sharp, R 226
Sheingold, K 248
Shelly, B 375
Sherwood, C 367
Sheviak, J 39
Skillen, P 127
Soloway, E 229
Somekh, B 246
Songer, N 351
Stager, G 18, 20
Stager, J 372
Stapleton, J 289
Stearns, P 378
Sterman, J 290
Stewart, L 356
Stock, M 300
Stone, A 144, 227
Strang, H 378
Sturdivant, P 392
Summers, M 282
Susskind, O 127
Swan, K 24
Sweeney, J 293
Taylor, H 40
Taylor, R 273
Teague, G 261
Teague, M 261
Tempel, M 20
Thomas, L 221
Tinto, P 375
Tucker, A 59
Turner, A J 59
Turner, S 141
Ullrich, J 94

Varden, S 65
Wagner, A 313
Walker, M 1
Walters, J 293
Warner, K 233
Warren, B 265
Watson, J A 22
Watt, D 180
Watt, M 180
Weinberger, D 313
Weintraub, H 391
Wepner, S 351
Weslow, G E 261
West, G 304
Wheeler, D 175
White, B 362
White, C 332
Whitelock, R 135
Wiebe, J 221
Willet, D 292
Williams, E 235
Withrow, F 146
Wolfe, J 314
Wood, A 144
Yoder, S 18
Zinck, R A 256
Zirkler, D 289
Zollman, D 336
Zorfass, J 58, 121

NECC '89 Connections
Lesley College
29 Everett Street
Cambridge, MA 02138-2790

Non-profit Organization
U.S. POSTAGE
PAID
Boston, MA
Permit No. 56840

PROCEEDINGS